La Methodologie d'acquisition des connaissances KADS et les explications

Philippe Martin

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

La méthodologie d'acquisition des connaissances KADS et les explications

Philippe MARTIN

N° 2179
Janvier 1994

PROGRAMME 3

Intelligence artificielle,
systèmes cognitifs et
interaction homme-machine

La méthodologie d'acquisition des connaissances KADS et les explications

Philippe MARTIN

E-mail : phmartin@sophia.inria.fr - Tél.: 93 65 76 45

Résumé : Afin d'aider à la construction de systèmes à base de connaissances (SBC) capables de fournir à leurs utilisateurs de bonnes explications sur leurs connaissances et raisonnements, ce rapport montre tout d'abord l'intérêt de prendre en compte les explications dès les phases d'acquisition des connaissances et de conception du système. Nous effectuons ainsi une synthèse sur :
- les différents types de méthodes actuellement utilisées pour acquérir et modéliser les connaissances d'un SBC ;
- la méthodologie KADS et notamment sa bibliothèque de modèles d'interprétation permettant de guider l'extraction et la modélisation des connaissances nécessaires à l'application ;
- les différents types ou rôles de l'explication ainsi que les techniques et stratégies explicatives actuellement employées ; nous proposons ensuite des spécifications pour les SBC explicatifs.

Puis nous étendons le Modèle Conceptuel de KADS pour permettre de modéliser des connaissances liées aux explications, autres que celles déjà modélisées pour la résolution de problèmes. Plus précisément, nous proposons d'inclure au modèle de coopération de KADS, un modèle d'expertise de coopération et un modèle d'expertise de communication. Nous spécifions ces deux nouveaux modèles d'expertises en décrivant leur type de contenu ainsi que les relations entre ces modèles. Nous offrons un guide pour aider à la construction du modèle d'expertise de communication ; nous proposons pour cela un modèle d'interprétation, avec en particulier des structures d'inférences commentées.
Nous complétons cette aide à l'extraction et à la modélisation des connaissances nécessaires aux explications, en fournissant une liste de questions pour acquérir connaissances opératoires et connaissances explicatives liées à chaque type d'élément d'un modèle d'interprétation.

Mots-clés : acquisition des connaissances, KADS, expertise explicative, expertise de coopération, modèle d'expertise, modèle d'interprétation.

Explanations and the KADS Knowledge Acquisition Methodology

Abstract : In order to help the design of Knowledge Based Systems (KBS) able to provide their users with good explanations on their knowledge and reasonings, this report first shows the interest to take into account explanations in the earlier phases of knowledge acquisition and system design. We present a synthesis of :
- the various methods currently used for acquiring and modelling the KBS knowledge ;
- the KADS methodology and in particular its library of interpretation models which enables to guide the extraction and modelling of the knowledge needed for the application;
- the various types or roles of explanation and the currently used explanatory techniques and strategies; we propose then specifications for explanatory KBS.

Then we extend the KADS Conceptual model in order to enable the modelling of explanation related knowledge. More precisely, we propose to include in the KADS model of cooperation, a model of cooperation expertise and a model of communication expertise. We specify these two new models of expertise by describing their content type as well as the relations between both models. Moreover we offer some guidelines for the design of the model of communication expertise ; thus we propose an adequate model of interpretation with in particular commented inference structures.
We complement this help to extraction and modelling of knowledge needed for explanation, by a list of questions enabling to acquire problem solving knowledge and explanatory knowledge related to each type of element of an interpretation model.

Keywords : knowledge acquisition, KADS, explanatory expertise, cooperation expertise, model of expertise, interpretation model.

SOMMAIRE

SOMMAIRE DE L'ANNEXE : Les tâches génériques de KADS

Introduction

Ce rapport présente une exploitation de la méthodologie d'acquisition des connaissances KADS pour aider à produire des systèmes experts ou systèmes à base de connaissances (SBC) capables de fournir de bonnes explications de leurs connaissances et raisonnements à leurs utilisateurs.

Pour cela, après avoir donné une première vision des explications, le premier chapitre met en évidence des spécifications sur les connaissances et l'architecture de ces SBC explicatifs. Il montre ainsi la nécessité de prendre en compte les explications dès la phase d'acquisition des connaissances et de conception du système.

Le chapitre 2 présente alors rapidement les différents types de méthodes actuellement utilisées pour acquérir et modéliser les connaissances et l'architecture d'un SBC. Puis, le chapitre 3 détaille la méthodologie KADS et notamment la phase de modélisation de l'expertise nécessaire à l'application. Il apparaît alors que les spécifications citées ci-dessus peuvent être prises en compte et constituent un avantage même pour les aspects non explicatifs du SBC.

Le chapitre 4 dresse un tableau des différents types ou rôles de l'explication ainsi que des techniques et stratégies explicatives actuellement employées. Des spécifications sur les connaissances d'explications (relatives à la construction de l'explication) sont ainsi données.

Compte-tenu de celles-ci, le chapitre 5 développe les modèles de coopération et de communication de la méthodologie KADS, ce qui permet d'offrir au concepteur qui utilise ces modèles, une aide ou un guide lors de la construction du module d'explication du SBC. Ces modèles sont semi-indépendants de la façon dont a été modélisée l'expertise liée à l'application. Celle-ci peut donc être modélisée par des méthodologies autres que KADS, quoique similaires, et en respectant les spécifications données au chapitre 1.

Enfin, le chapitre 6 offre une aide dans le cadre de KADS pour l'acquisition des connaissances nécessaires à l'application et aux explications. C'est une sorte d'aide mémoire permettant de modéliser de façon la plus complète et cohérente possible ces connaissances.

L'annexe est une présentation détaillée (et en français !) des modèles (un par classe de problèmes) offerts par KADS pour aider le concepteur à acquérir et modéliser l'expertise nécessaire à son application. Elle unifie dans un même cadre des descriptions diverses (différents auteurs ou évolutions des modèles).

Les chapitres les plus novateurs sont donc le 5 et le 6. Les autres sont plutôt des chapitres de synthèse de domaines très riches en directions de recherches et opinions complémentaires ou concurrentes. Ce rapport cite de multiples possibilités de fertilisation croisée entre ces domaines et en développe une.

Chapitre 1 Des connaissances structurées pour les explications

1 Une première vision des explications

Expliquer peut avoir des significations très différentes. Voici une première distinction :
- expliquer un phénomène (physique, social, état du système, etc.) au sens de l'explication scientifique i.e. pour le comprendre; ceci relève des méthodes de résolution de problème ou de l'apprentissage;
- donner à un interlocuteur la signification ou le caractère de validité d'un fait ou d'une proposition, cet objet de l'explication étant déjà connu par le producteur de l'explication (suite à des observations, un raisonnement, etc.); exemple : justifier un choix.

Dans ce rapport, c'est le deuxième cas qui est traité, l'agent explicateur étant une machine.
Pour fournir une explication, un agent explicateur (humain ou machine) doit interpréter une requête (ou demande d'explication) puis y répondre en fournissant une analyse (définition, analogie, statistique, etc.) ou un rapport causal (motivation, enchaînement logique, etc.) entre des faits connus (du récepteur) et le fait sur lequel porte la requête (ex: entre des symptômes et une maladie précise). Ce produit, que l'on appellera discours ou explication, peut s'exprimer sous diverses formes : langue naturelle, représentations graphiques, simulations, etc.

En informatique classique, les explications se réduisent à des commandes d'aide interactive qui présentent les fonctionnalités des programmes et la manière de les utiliser.
Mais les comportements de ces derniers sont rarement détaillés : interprétation des données, action en cours, méthode ou stratégie utilisée pour arriver aux résultats. Pourtant, dans le cas des systèmes complexes ou d'aide à la décision, ayant des sources de connaissances multiples ou incertaines et des contrôles de raisonnements souvent élaborés, ce type d'explication (justification des décisions et des résultats) est indispensable.
C'est le cas des Systèmes Experts (SE) ou Systèmes à Bases de connaissances (SBC)[1], cadre dans lequel se situe ce rapport, et où les explications sont nécessaires [Dieng 91] :

- à l'utilisateur final (expert, ingénieur, utilisateur non spécialiste) qui hésite moins à suivre un «conseil» s'il peut avoir accés au raisonnement et aux heuristiques utilisées (aux choix effectués et aux justifications de ces choix). Elles peuvent également lui servir à faire un choix entre plusieurs alternatives, etc. ;
- à l'expert qui peut ainsi vérifier si ses connaissances ont été bien transmises et si les heuristiques introduites pour accélérer le raisonnement n'engendrent pas d'erreurs;
- aux autres experts pouvant, à travers les explications, accéder à l'expertise du premier expert, ce qui peut faciliter une multi-expertise ou aider à valider les connaissances du SBC;

1. Les SBC ont l'architecture des SE (base de connaissances, moteur d'inférences), mais ne s'emploient pas seulement pour reproduire une expertise. Les deux termes seront employés indifféremment dans ce rapport

- pour des étudiants, les SBC se prêtant bien à l'enseignement quand ils possèdent une masse importante de connaissances à la fois structurées et séparées des connaissances de contrôle; en Enseignement Intelligemment Assisté par Ordinateur, par exemple, ils doivent entraîner leur utilisateur par des exercices, corriger ses erreurs de façon appropriée, lui donner des indices pour démarrer, etc.
- au concepteur du SBC qui peut déterminer la validité de son système et les raisons de ses défaillances (base de connaissances (BC) ou moteur d'inférence); c'est l'un des rares cas où les explications doivent refléter précisément les différentes étapes du raisonnement en suivant l'implémentation pour mettre en évidence les connaissances correctement utilisées.

Le contenu des explications doit donc clairement tenir compte du type d'utilisateur et de ses attentes. Les traces de raisonnement et les textes pré-enregistrés, bien que possédant de nombreuses informations, ne sont pas forcément explicatifs. Un système de diagnostic doit par exemple convaincre son usager de la pertinence de ses questions et de ses prescriptions.

La valeur explicative d'une réponse est accordée par l'interlocuteur en fonction de ses besoins et connaissances, lesquels ne peuvent être clairement déterminés à partir de la demande d'explications : plusieurs réponses sont alors possibles, et pour choisir la plus adaptée, il faut faire des hypothèses sur ce que cherche l'utilisateur ([Carcagno 92]).
Pour valider ces hypothèses, il faut se doter d'un «modèle de l'utilisateur» très précis pour tenter de trouver la bonne explication du premier coup (cf par exemple [Weiner 89]) ou/et instaurer un dialogue entre le sytème et l'utilisateur jusqu'à ce qu'il soit satisfait.
Le modèle de l'utilisateur (personnel si possible) est soit rentré par ce dernier, soit déduit de ses questions ou comportements et capable d'évoluer au cours de consultations successives. Il comprend son niveau d'expertise, ses buts et, tout particulièrement en enseignement, son modèle (supposé) du domaine et de résolution de problème.

Une fois les informations de l'explication recueillies, il faut les présenter de manière appropriée à l'utilisateur : textes, hypertextes, graphiques (statistiques, visualisation de l'arbre de raisonnement, ...), etc.
La production de discours s'inspire donc des travaux en sciences cognitives.
Chez les hommes, la stratégie du discours la plus courante est la négociation ([Schank 73]) mais ceci est encore faiblement exploité pour les explications qui font plutôt appel aux techniques de planification de textes (ou de discours) : grammaires d'explications de Cawsey, planification de Moore, relations rhétoriques, etc.

Cet aspect communication de l'explication sera détaillé dans les chapitres 4 et 5. Nous allons en effet suivre la classique division des recherches en explications dans les SBC.

- La première direction consiste à travailler au niveau de la représentation des connaissances à expliquer. L'idée de base de cette approche est la suivante : plus les connaissances à expliquer seront explicitées et représentées clairement, plus la tâche du module d'explication en sera facilité. Explicable est alors synomyme d'**explicite**.
- La seconde s'occupe de la production d'explications et complète donc la première approche : identification des éléments pertinents, mise en forme, adaptation au contexte, choix entre différentes explications potentielles, etc. Il s'agit là de l'explication en tant que **discours**.
  Il faut ensuite transférer ce discours à l'utilisateur et donc prendre en compte les aspects interactifs de ce transfert. On réfère ici à l'explication en tant que **dialogue**.
  Cette seconde direction s'attache donc à transformer un contenu (trace de raisonnement, hiérarchie de concepts, etc.) donné dans une certaine représentation, en une explication dépendante du contexte d'interaction avec l'utilisateur (médium, historique du dialogue, etc.).

Nous allons donc maintenant donner quelques spécifications sur l'architecture des SBC explicatifs (i.e. ceux qui outre leur fonction de résolveur de problèmes, d'exploitation d'une base de connaissances, ont une fonction d'explication de leurs résultats ou démarche suivie).

La synthèse de ces spécifications dans le cadre de KADS sera faite au chapitre 5.

Pour cela, il faut savoir quels sont les différents types d'explications. La section suivante donne les principaux et les plus utilisés. Cette typologie sera plus utilement complétée au chapitre 4.

2 *Types d'explications*

Avant de donner la classification de [Chandrasekaran 89], voici une pré-liste rapide de ce que peut faire un SBC ([Dieng 87a]) :

- analyser la trace du raisonnement et la BC (base de connaissances) compte-tenu de la spécification du problème en disposant de plusieurs *filtres* : modèle de l'utilisateur, degré d'approfondissement désiré, entité intéressantes (en fonction du contexte), etc. La même entité (objet, états ou ensemble d'états lors du raisonnement, etc.) doit donc pouvoir être vue de plusieurs points de vue différents suivant le type et les désirs de l'utilisateur
- offrir des informations descriptives sur les entités manipulées : objets, règles, stratégie ou plan de travail, liens structurels ou fonctionnels reliant ces différents éléments (réseau de dépendances, paquets de règles, ...), etc;
- faire le bilan d'un raisonnement passé : historique détaillé, résumé ou synthèse, morceaux choisis. Ceci par exemple pour justifier un résultat ou aider à corriger la BC pour l'expert ou le cogniticien (personne chargée du recueil de la connaissance) et le moteur d'inférences pour le progammeur;
- proposer des prévisions en cours de raisonnement et surtout, adapter toutes ces possibilités au type de l'utilisateur (besoins, degré de détail désiré, etc.).

Chandrasekaran distingue au moins trois types d'explications de SBC [Chandrasekaran 89] :

- Type 1 (ou «règle») : description d'une règle (au sens large de connaissances opératoires) ou de la chaîne de déduction (règle(s), faits, etc) ayant permis de déduire certains faits et avoir pris (ou pas) certaines décisions. Ainsi peut être donnée l'origine d'un fait : soit en l'extrayant de la base de connaissances, soit en exploitant l'arbre des déductions (ou trace).
  Ce sont les seules explications que peuvent fournir les SE de première génération (SE1) car elles n'exigent pas la représentation explicite des «principes de base» du domaine, ni celle du comportement du SE en termes de tâches et de stratégie (exemples et détails section suivante);

- Type 2 (ou «BC») : explication sur les éléments de la BC ou pourquoi ils sont utilisés lors d'un raisonnement. Il existe au moins quatre moyens de **justifier** une connaissance : par référence à des sources (textes, livres, spécialistes), en généralisant à partir d'exemples («un patient qui avait des symptômes similaires était atteint du ...», «dans 68% des cas, ceci marche»), par inférence (résolution du problème puis explicitation comme au type 1) ou encore en référant aux connaissances profondes (cf encadré 1) dont elle dérive.
  La méthode «par inférence» peut aussi faire appel à des connaissances profondes (CP) pour

expliciter la nature (sémantique, certitude, etc.) du lien déductif existant entre les conditions et la conclusion d'une (suite de) règle(s)). Plus couramment, justifier une connaissance par des CP consiste à la décomposer de façon à faire apparaître les étapes intermédiaires d'un raisonnement compilé ([Clancey 83]) ou bien les principes du domaine ayant permis de l'élaborer (détails section suivante).
Cependant, la justification ne doit pas énoncer le processus complet sous-jacent à une connaissance experte mais permettre d'abstraire ou de généraliser ce processus afin d'établir un lien avec un schéma général déjà connu ([Reynaud 91]). Pour une même connaissance, il est par ailleurs intéressant de pouvoir fournir plusieurs justifications se rapportant chacune à un point de vue particulier.

- Type 3 (ou «stratégie») : des explications sur le contrôle et la stratégie employée lors de la résolution. La BC doit donc comporter un niveau «tâche» où ces informations sont explicites. Ainsi NEOMYCIN [Clancey 84], qui résout les mêmes problèmes de diagnostic que MYCIN, contient des opérateurs de diagnostic tels «établir les hypothèses» ou «explorer et affiner» à travers lesquels s'opère et peut s'exprimer la stratégie de diagnostic.
Ces explications sur la stratégie d'une tâche peuvent être données en élagant ou approfondissant au gré de l'utilisateur un arbre des tâches.
Il est alors aussi possible de donner des explications «négatives» i.e. sur un résultat non fourni car impossible à déduire d'après la base de faits ou écarté à la suite d'une décision lors de la résolution (cette décision n'est explicable que si le contrôle et la stratégie sont explicites).

A titre d'exemple de ces types d'explications, voici un dialogue (fictif) Usager - Système (extrait de [Bouri 90b]) :

```
U: Pourquoi la réduction d'impôt est une bonne manière de réduire le déficit commercial ?
S: Car elle encourage l'épargne, stimule l'investissement et accroît la production (ce qui
fait décroître les prix, accroître les exportations, rend les marchandises attractives et
réduit le déficit commercial) [Type 2].
U: Pourquoi ne suggérez-vous pas d'accroître les tarifs douaniers ?
S: Ceci implique des coûts politiques [Type 1] et ma statégie est d'abord de faciliter les
plans politiques [Type 3].
...
U: Comment avez-vous obtenu la valeur de l'attribut A ?
S: Je dispose de trois méthodes pour inférer la valeur de A; j'utilise dans l'ordre :
   1) déduction à partir de la base de règles BR1
   2) accès à la base de données historique BD1
   3) question à l'utilisateur.
  J'ai utilisé la base de règles BR1, mais aucune valeur n'a pu être déduite. J'ai consulté
  la base de données et j'y ai trouvé la valeur V pour A [Type 3].
```

La plupart des SBC explicatifs (numériques mis à part) utilisent au moins un de ces trois types d'explication. Construits pour pallier aux insuffisances des SE1, les **SE2** (seconde génération) permettent d'offrir l'**ensemble** de ces possibilités : les «Connaissances Utiles pour les Explications» de type 2 (i.e. buts, fonctionnalités, relations avec les autres entités, etc.) peuvent être stockées dans les modèles statiques du domaine (donc dès la construction de la BC); ils disposent également d'autres niveaux pour exprimer la stratégie à employer, notamment en termes de tâches.

3 *Insuffisances des SE1 pour les explications*

Quoique les traces simples soient plus puissantes dans un SE que dans un logiciel classique (à cause de la séparation entre la BC et le moteur d'inférences, de l'interactivité du système et du caractère relativement significatif pour l'usager de la règle imprimée), elles sont peu explicatives, l'utilisateur devant reconstituer manuellement l'enchaînement des règles.
Le SE doit donc générer une trace sélective ou «*arborescence des buts*» mémorisant les différentes étapes du raisonnement et leurs enchaînements. Chaque noeud de l'arbre contient des informations sur un des buts envisagés, les actions entreprises pour l'accomplir (règles ou questions à l'utilisateur) et leurs résultats. Cet historique est ensuite exploité pour un module d'explications ou d'apprentissage. Davis l'implante en 1982 dans le 1er SE à avoir des possibilités d'expliquer son raisonnement : Mycin (Shortliffe 73). Voici un exemple de dialogue Généraliste - Mycin cité dans [Bouri 90a] :

```
G: Règle 184.
M: SI la catégorie de l'organisme est connue ET la coloration de Gram de l'organisme
   est gram- ET la morphologie de l'organisme est de type bâtonnet ET l'aérobie de
   l'organisme est facultative ALORS il existe une évidence faiblement suggestive
   (degré 10%) que l'identité de l'organisme soit pseudomanas-aeriginosa.
...
M: L'infection du malade a-t-elle été contractée à l'hôpital ?
G: Pourquoi ?
M: Pourquoi est-ce important de le savoir ? Cela aidera à déterminer la catégorie à
   laquelle appartient l'organisme responsable de l'infection. Car il a été établi
   que 1) l'organisme est à gram positif, 2) la morphologie de l'organisme est
   batonnet, 3) l'aérobicité est facultative
   donc si : 4) l'infection n'a pas été contractée à l'hôpital, alors il est presque
   certain (degré 80%) que l'organisme est l'enterobacteriaceae; de plus il se peut
   (degré 10%) que ce soit le pseudomanas aeruginosa.
G: Comment le 3 ?
M: Comment a-t-il été établi que l'aérobicité de l'organisme est facultative ? Les
   arguments suivants ont été employés: 1) la règle 027 a indiqué qu'il existe une
   forte évidence (degré 80%) pour que l'aérobicité soit facultative, 2) la règle
   027 a indiqué qu'il existe une faible évidence pour que l'organisme soit
   anaérobique. Il a donc été établi que l'aérobicité de l'organisme est facultative.
G: Comment le 1 ?
M: Comment a-t-il été établi que le gram de l'organisme est négatif ? Parce que vous
   l'avez dit en réponse à la question 11.
G: Pourquoi n'avez-vous pas envisagé la possibilité que l'organisme soit un
   streptocoque ?
...
```

Encadré 1. Premières explications (Mycin) et premier type d'explications

L'architecture d'un SE, son mode de raisonnement, l'organisation et la conceptualisation de sa BC ainsi que les formalismes de représentation des connaissances utilisés ont une influence directe sur le type d'explication qu'il est capable de fournir.

Les SE1 permettent une résolution très rapide des problèmes car leur raisonnement est essentiellement fondé sur des connaissances «compilées» [Chandrasekaran 83] (également appelées «de surface» ou «empiriques» ou encore «expérimentales» par d'autres auteurs). Mais ces connaissances sont difficiles à obtenir (car connues seulement de l'expert et souvent inconsciemment assimilées d'où des oublis, des déformations, etc.), difficilement justifiables (si empiriques), non structurées (principes du domaines et heuristiques données en vrac) et difficilement interprétable par un non-initié du domaine (le cogniticien ne comprend généralement pas les références implicites aux schémas de connaissances structurés connus).

Aussi, les explications des SE1 (exemple dans encadré 1), basées sur un filtrage de la trace du raisonnement ne sont pas satisfaisantes : seule l'information présente dans le code pouvant être paraphrasée, le système ne peut justifier son raisonnement ni le pourquoi des connaissances utilisées. Ex: la règle «Si le patient a moins de huit ans, alors ne pas prescrire de tétracycline» est une optimisation de plusieurs règles «la tétracycline se dépose sur les os en formation», elle «change la teinte des dents», etc.
Ces explications reflètent les actions réellement effectuées (et peuvent donc servir pour certaines mises au point) et dépendent fortement de la manière dont a été écrit le code (l'explication ne sera pas structurée si le code ne l'est pas).

Le formalisme de représentation des connaissances utilisé dans les SE1 (à base de règles ou de réseaux sémantiques), du fait de sa simplicité et de son uniformité, semblait à l'origine adéquat tant pour le raisonnement que pour l'explication. Cependant, il s'avéra vite de trop bas niveau d'abstraction dans les deux cas. Exemple cité dans [Bouri 90b] : le ratio `«rentabilité des capitaux = résultat net / capitaux permanents»` peut se voir représenté par `«SI capitaux-permanents-stables ET résultat-net-augmente ALORS rentabilité-des-capitaux-augmente».`
Il y a perte d'information :
- cette règle n'est qu'une interprétation possible du ratio et n'est plus reliée à ce concept;
- les concepts atomiques (avec des «-» à l'intérieur) sont indécomposables, d'où des explications lourdes, syntaxiquement incorrectes, superficielles ou impossibles (ex: qu'appelle-t-on «résultat net» ou «quand les capitaux sont ils qualifiés de «permanent»).

Autre conséquence du manque de structuration des formalismes utilisés, le contrôle et la notion de tâche y sont implicites. Ainsi, Mycin est incapable d'expliquer sa stratégie, celle-ci étant contenue implicitement dans l'ordre des prémisses des règles. Plus généralement, un SE de diagnostic, organisé autour d'une architecture à base de règles, pourra difficilement répondre aux questions [Dieng 87a] «pourquoi telle hypothèse n'a pas été considérée ?» ou «l'hypothèse retenue rend-t-elle compte de tous les symptômes ?» parce que les concepts «symptôme», «hypothèse de diagnostic» lui sont inconnus et la tâche «considérer une hypothèse de diagnostic» ne correspond pas à celles de plus bas niveau qu'il emploie : «invoquer une règle», «évaluer sa prémisse», etc.

Le fondement des connaissances et les principes de haut niveau du raisonnement, appelés connaissances profondes ou complètes, sont avantageux tant pour les explications («Connaissances Utiles pour les Explications») que pour le développement et la maintenance d'un SE (clarté, généricité, abstraction). La prise en compte de nombreux types de connaissances (heuristiques, définitions, modèles causaux, tâches, métarègles explicatives ou de stratégie, etc.) et sous différentes représentations (floues, temporelles, règles, objets, agents, etc.) ont donné naissance aux SE2 (systèmes experts de seconde génération).

Décrire l'activité du SE grâce à un arbre de tâches facilite l'expression de la stratégie et permet un niveau d'abstraction élevé. C'est un des grands thèmes de recherches des SE2 et donc des explications. On peut aussi, comme [Chandrasekaran 89] le conseille, associer des mécanismes d'explications aux tâches génériques, celles-ci étant des abstractions de tâches (comme la classification hiérarchique, la conception par sélection de plan et raffinement, l'héritage de propriétés, etc.) servant de modèles pour construire les tâches du SE. Ce point étant détaillé plus loin, ainsi que celui de l'apport de l'acquisition des connaissances sur les explications, faisons (hors ces deux sujets) un rapide bilan des recherches sur les explications dans les SBC.

4 *Recherches sur les architectures explicatives*

Historiquement, diverses voies ont été explorées pour construire des SE explicatifs. Les premières idées accompagnent les progrés des SE.

- Utiliser de multiples représentation des connaissances, ACE [Sleeman 77].
- Extraire des connaissances sur les connaissances, NEOMYCIN [Hashing 83].
- Dissocier la ligne du raisonnement de celles de l'explication : la tâche d'explication est une tâche à part entière raisonnant sur la trace du SE afin d'en montrer les étapes importantes, la justifier, la résumer ou l'adapter au niveau de l'interlocuteur.
  Dans cet approche, on peut citer CQFE [Kassel 86] qui utilise un modèle conceptuel des objets du domaine pour raisonner sur le raisonnement et le système Pourquoi-Pas [Safar 89] qui compare le raisonnement du système avec celui (supposé) de l'utilisateur afin d'expliquer pourquoi un résultat attendu par ce dernier n'a pas été déduit.
  Il est également possible de construire une preuve indépendamment du processus mis en oeuvre dans la résolution, de la même façon qu'une démonstration mathématique prouve un théorème sans rappeler comment le mathématicien l'a trouvée. Ceci est utile lorsque la trace est insuffisante pour être exploitée ou ne peut être rendue compréhensible de l'interlocuteur (cf [Wick 92]).

Ces idées sont maintenant reprises par la plupart des SE explicatifs.

Les idées suivantes sont plus modernes et plus élaborées.

- Disposer d'un générateur de programmes instanciant sur le domaine de l'application des principes de plus haut niveau, et mémoriser le processus de cette génération du programme. Ainsi, bénéficie-t-on à la fois d'un code (ou règles au sens large) adapté au domaine donc rapide (règles «compilées») et d'un moyen de retrouver les principes qui ont permis de le générer. Ceci permet d'acquérir auprès de l'expert des connaissances de haut niveau (donc mieux modéliser l'expertise) et d'obtenir des explications suffisamment abstraites.
  [Swartout 83] avec son système XPLAIN est à l'origine de cette méthode de correspondance entre connaissances opératoires et connaissances profondes.
- Construire un SE «explicateur» observant le SE résolvant le problème et raisonnant sur les actions de ce dernier. Dans [Dieng 87b], la représentation objet est utilisée pour avoir plusieurs niveaux d'explication (explications éventuellement automatiques par un système proche des valeurs actives);
- Disposer d'une grande quantité de connaissances permettant de résoudre le problème mais aussi de comprendre les actions effectuées et les choix écartés lors de la déduction [Dominguez 89].

Tout ceci conduit donc à constater que les explications doivent se préparer dès la phase d'acquisition des connaissances et de modélisation du système. La prise en compte des explications pose en effet des contraintes sur l'architecture du système, la formalisation de ses connaissances.
Dans ce rapport, les **«connaissances explicatives»** désignent toutes les **«Connaissances Utiles pour les Explications»** mais inutile pour l'application (**connaissances opératoires** ou de résolution). Les chapitres 4 et 5 détaillent ces connaissances explicatives (sans revenir sur les connaissances profondes qui en font aussi partie). Ces deux types de connaissances (explicatives et opératoires) étant liés (ceci sera détaillé), le chapitre 6 montre comment l'on peut les acquérir en même temps.

En résumé, un SBC explicatif doit disposer [Dieng 87a] :

- de plusieurs *niveaux de granularité* (règles ou concepts, paquet ou classe de règles, tâche à résoudre, stratégie, connaissances profondes, etc)
- de plusieurs points de vue sur les divers objets qu'il manipule : connaissances, états ou ensemble d'états lors du raisonnement, etc.
- de *filtres* pour exploiter ces diffèrents points de vue et ce, en fonction : du modèle de l'utilisateur, du degré d'approfondissement qu'il désire, des entités jugées intéressantes (en fonction du contexte), etc.
  Par exemple, l'arbre de raisonnement doit pouvoir être compacté selon différents critères (parties linéaires, etc.) et les noeuds importants de l'arbre de raisonnement doivent pour cela être étiquetés et annotés (propriétés importantes, spécifications sur la suite du travail, etc.).

Si pour de bonnes explications, il est indispensable d'acquérir des connaissances bien structurées, cela est de toute façon apparu également très important pour obtenir un code (de résolution) puissant et maintenable. Les deux chapitres prochains détaillent plus précisément différentes manières d'obtenir, de modéliser et de coder ces connaissances, avec leurs avantages et leurs inconvénients.

Chapitre 2 L'acquisition des connaissances

1 Le transfert d'expertise

L'ingéniérie de la connaissance intéresse de plus en plus le monde industriel et commercial en offrant un moyen de résoudre des problèmes spécifiques relatifs à la gestion des savoir-faire d'une entreprise [Brunet 91].

La conception d'un système expert exige un travail de transfert de connaissances entre des sources d'expertise (experts humains ou documents) et un outil informatique de façon à disposer ensuite d'un SBC pouvant être consulté comme un expert. Le recueil de la connaissance et sa traduction dans un formalisme intermédiaire ou final (celui du générateur de systèmes experts cible) est effectué par un ou des ingénieurs de la connaissance (ou cogniticiens).

Cette phase de *transfert d'expertise* est souvent considérée comme le goulot d'étranglement du développement d'un système expert. En effet, elle peut durer des mois voire des années et, compte-tenu de la complexité des connaissances de l'expert qui parvient difficilement à expliciter ses processus mentaux, la connaissance extraite risque souvent d'être inexacte, incomplète, voire inconsistante ([Dieng 90b]). De nombreuses techniques de verbalisation et d'aide aux interviews ont été développées pour "extraire" les connaissances subconscientes ou implicites de l'expert [Hoffman 89, Aussenac 89]. Elles sont souvent inspirées des travaux en psychologie cognitive. Notons:
- les interviews, généralement enregistrés et souvent non dirigés au début de l'extraction pour ne pas introduire de biais dans celle-ci;
- l'analyse de protocole verbaux où l'expert résoud un problème en réfléchissant à haute voix;
- l'introspection, l'expert dit comment il résoudrait un cas imaginaire typique (peu fiable car les stratégies et modes de raisonnements énoncés ne sont pas nécessairement ceux utilisés);
- l'observation directe où l'expert est filmé en situation de travail;
- la simulation du comportement du futur SE par l'expert;
- les commentaires sur des protocoles antérieurs (reprise d'un cas déjà traité);
- le tri par l'expert des problèmes suivant leurs degrés de difficulté, le type de résolution attachées, etc. Cette technique, facile et peu couteuse, permet de découvrir des relations entre concepts;
- l'utilisation d'un questionnaire, une fois qu'une partie de l'expertse a été identifiée;
- le «brainwriting» pour regrouper et comparer les idées de plusieurs spécialistes : chacun enrichit les idées des autres en les commentant par écrit (ex: une feuille par idée).

Dans chaque technique, il est essentiel de minimiser les biais i.e. les déformations de l'expertise. Cependant les données sont souvent incomplètes (omissions, connaissances «compilées» ou supposée à tort connues, manque de coopération ou difficulté de la connaissance être verbalisée) ou inexacte. La section suivante montre que pour interpréter et exploiter ces données, deux philosophies co-existent.

2 Cycle de vie

De ces deux philosophies de développement des SBC, la première est maintenant moins utilisée. Il s'agit du *prototypage rapide* [Hayes-Roth 83, Harmon 85] qui consiste à développer après une étape de définition du problème et un nombre suffisant mais minimum d'entretiens une maquette (en quelques mois) : conceptualisation (avec l'optique des contraintes d'implémentation), formalisation, implantation et tests. Ensuite, soit elle est raffinée grâce à de nouvelles interviews, ce qui entraîne un certain nombre de retours-arrière, soit la technique utilisée s'avère ne pas convenir et une autre maquette est développée avec un autre formalisme de représentation des connaissances et d'autres mécanismes d'inférences (ex: règles, frames, blackboard, etc.). L'insuffisance d'informations structurelles (peu de modélisation) et l'utilisation de langages plus ou moins universels (lisp, prolog, etc.) rend en effet le choix de la représentation un peu aléatoire (d'où l'intérêt du développement rapide d'une maquette). Cette méthode d'acquisition est dite «dirigée par l'implémentation».
Le système complet est développé par extension ou modification totale de la base de connaissances de la maquette et validé par rapport à des bibliothèques de cas tests ou par d'autres experts que ceux ayant participé à son développement. Enfin, il est intégré dans l'entreprise (avec une éventuelle connexion à d'autres logiciels) et maintenu.

Dans l'*acquisition structurée des connaissances*, le problème est défini, puis l'expertise recueillie est **modélisée** avant de concevoir le SE puis de l'implanter, valider et intégrer comme ci-dessus. L'existence d'un modèle intermédiaire indépendant des contraintes d'implémentation permet au cogniticien de mieux exprimer l'expertise. En effet, pour utiliser les nombreuses données recueillies, il doit les interpréter (en fonction de buts, de leurs futures utilisations), les **organiser** (ou structurer), **abstraire** des concepts, **en un mot**, construire un **modèle**; d'autant qu'il ne s'agit pas de refléter des processus réellement mis en oeuvre dans le cerveau humain. Il ne s'agit donc pas en réalité d'un problème d'extraction mais bien de **réorganisation** des connaissances, de **reconstruction** des méthodes de résolution [Krivine 91].
Lors de la phase de conception, toutes les décisions techniques sont encore à prendre, mais le concepteur (qui peut alors être différent du cogniticien) dispose d'un modèle de haut niveau exprimant ce que le SE final peut et doit faire. Ceci est très utile pour le développement, la documentation, la mise au point, la maintenance ou le changement de technique d'implémentation. Il n'y a normalement pas (ou peu) de retours arrière vers le modèle intermédiaire (appelé dans KADS, le Modèle Conceptuel) pour ne pas biaiser la connaissance n'est pas biaisée par la vue de l'implémentation.
Ce Modèle Conceptuel est d'autant plus intéressant qu'aucun des formalismes plus ou moins universels (règles de production, logique, objets, etc.) connus aujourd'hui ne peut être utilisé pour *transcrire* l'expertise [Krivine 91] : de trop bas niveau d'abstraction, ils ne permettent pas de représenter la connaissance profonde de l'expert.

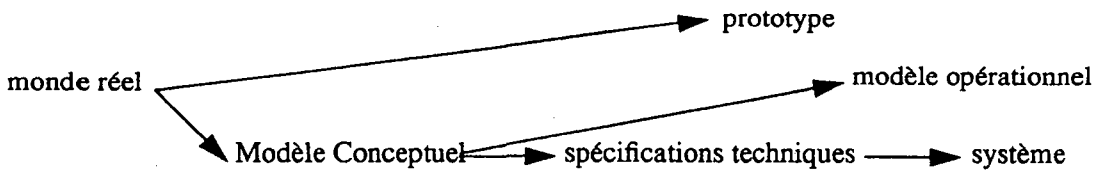


Figure 1 : Descriptions produites par les approches prototypage rapide / modélisation ([Voss 90])

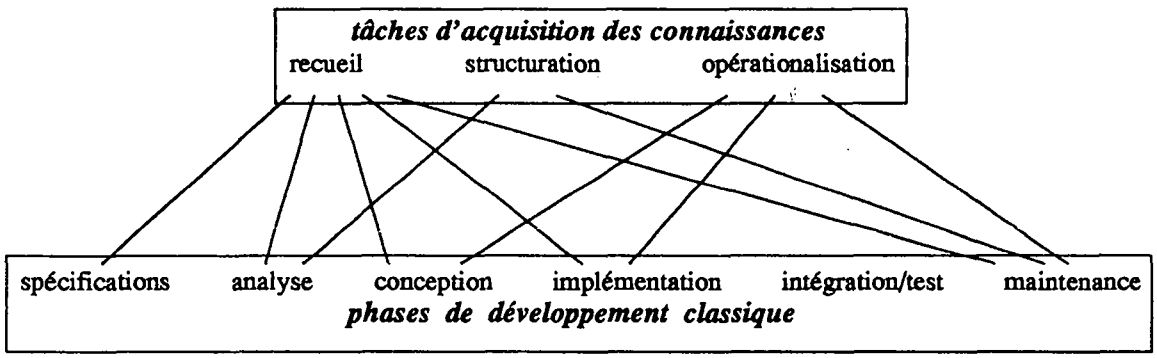
Les outils et méthodes proposés pour aider au transfert d'expertise et à la conception du SE tiennent compte de la spécificité des SBC tout en intégrant des apports d'autres disciplines : génie logiciel (KADS [Weilinga 92]), psychologie cognitive (KOD [Vogel 88] et MACAO [Aussenac 89]), linguistique et anthropologie cognitive (KOD) ou encore les *techniques inductives*

permettant d'apprendre de nouvelles connaissances à partir d'exemples [Quinlan 87, Kodratoff 87, Ganascia 89, Charniak 85].

3 *Les apports d'autres disciplines*

L'acquisition des connaissances relève de nombreux domaines.

- Un projet SBC est un projet informatique et doit donc bénéficier des techniques et recherches en génie logiciel pour assurer la qualité du développement : cycle de vie, spécification formelle, tests et validation. Ainsi, les méthodes KADS et KOD d'acquisition de connaissances et de développement de SBC ont des cycles de vie rappelant celui d'un logiciel classique : étude des besoins, analyse, conception globale, conception détaillée, codage et mise au point, tests, intégration et maintenance. Toutefois les phases de développement d'un SBC, cycliques par nature, ne peuvent vraiment concider avec celles des développements classiques. Voici un exemple de correspondance [Voss 90].



- Modèles mentaux, représentation de la connaissance, mécanismes du raisonnement, explications sur celui-ci et extractions d'informations adéquates à partir de celles-ci sont des sujets d'études de la psychologie cognitive. Ainsi MACAO, méthode et outil complet de développement de SBC, accessible au cogniticien comme à l'expert, est-il basé sur le modèle cognitif (de l'expert) comprenant des schémas empiriques (ou scripts) de résolution de problème et des schémas conceptuels plus abstraits, sorte de méta-connaissance sur les scripts, permettant de les utiliser suivant la stratégie imposée par le contexte.
  La méthode KOD propose également un modèle cognitif comprenant des taxinomies (description hiérarchique du monde physique), des actinomies (schémas mentaux d'actions type heuristiques) et des schémas d'interprétation (type connaissances profondes) permettant de comprendre le monde réel (relations causales) et de savoir quand déclencher les actinomies (stratégies). En amont de ce modèle dans le cycle de vie de KOD, se trouve le modèle pratique qui regroupe les manifestations verbales des trois paradigmes ci-dessus (description, action, déclaration). En aval, se situe le modèle informatique, sorte de conception détaillée, également structurée suivant les trois paradigmes.
- L'acquisition des connaissances et les processus de transferts dans une société humaine sont complexes et mal compris. Les neurosciences, la psychologie, la linguistique, l'éducation, la sociologie, l'anthropologie, la philosophie et la théorie des sytèmes sont autant de disciplines qui peuvent apporter des contributions significatives, à condition de les intégrer, d'adapter leurs vocabulaires spécifiques et de combiner leurs différents objectifs.

4 Directions de recherche actuelles

L'acquisition des connaissances, nous l'avons vu est à la croisée de nombreuses disciplines. Ses axes de recherches sont donc nombreux et divers [Dieng 90b] :

- les outils de seconde génération (coopération de multiples formalismes de représentation des connaissances et de raisonnement (heuristique, qualitatif, à base de modèles, etc.);
- l'extraction à partir de multiples experts i.e comment déceler consensus, conflits, correspondances et contrastes. Ex: architecture multi-agent, «brainwriting», etc.;
- la validation des SBC; ex: consistance et complétude du logiciel (par rapport aux spécifications) à chaque étape du cycle de vie; qualité, utilité et employabilité de la BC, etc.;
- l'intégration de l'apprentissage (par analogie, induction, à partir d'exemples, de contre-exemples, d'explications, etc.);
- l'intégration de l'analyse de textes permettant de décrire des élements de connaissance en langage naturel (ex: KALEX [Schmidt 89]);
- l'utilisation des hypermédias; l'expert peut alors entrer ses données librement puis les annoter avec des liens croisés. C'est le cas dans K-STATION [EC2] et SHELLEY [Anjewierden 92], respectivement support de KOD et de KADS;
- l'application des techniques d'extractions de la connaissance aux les tuteurs intelligents;
- la prise en compte de l'intégration dans le futur environnement du SBC;
- l'ergonomie et la prise en compte des interfaces utilisateurs (seule la connaissance nécessaire à l'application doit alors être extraite);
- la prise en compte de l'assistance à l'utilisateur (explications, etc.) et de la maintenance ultérieure (laquelle est facilitée par les explications). Ce thème est développé dans la section suivante;
- la prise en compte de la documention i.e le document de préétude, les documents produits lors de la réalisation du système et la documentation finale du SE (manuel de référence et manuel de l'utilisateur).

Le chapitre suivant développe la méthodologie KADS et précise ainsi la voie «modélisation» de l'acquisition des connaissances.

Chapitre 3 La méthodologie KADS

1 Cycle de vie

KADS[1] représente à l'heure actuelle la réponse la plus avancée en matière de méthodologie de développement de SBC [Campbell 89]. Synthèse et amélioration de nombreuses techniques, elle permet de traiter tout le processus d'acquisition des connaissances, du recueil au développement d'un système complet. C'est une méthode dirigée par les modèles (par opposition aux méthodes dirigées par l'implémentation). Les modèles de KADS étant des *représentations intermédiaires* partielles et orientées de l'expertise avant leur implantation dans un système ([De Greef 85]).

KADS propose un cycle de vie basé sur les méthodes de développement de logiciels. Une analyse complète des données précède la conception et l'implantation du SE, ce qui diffère donc du prototypage rapide ou des approches incrémentales. Les résultats de la phase d'analyse (contraintes externes, Modèle Conceptuel des objets du domaine et des méthodes de résolution) servent d'entrée à la phase de conception où ils sont transformés en spécification de l'architecture fonctionnelle du SBC.

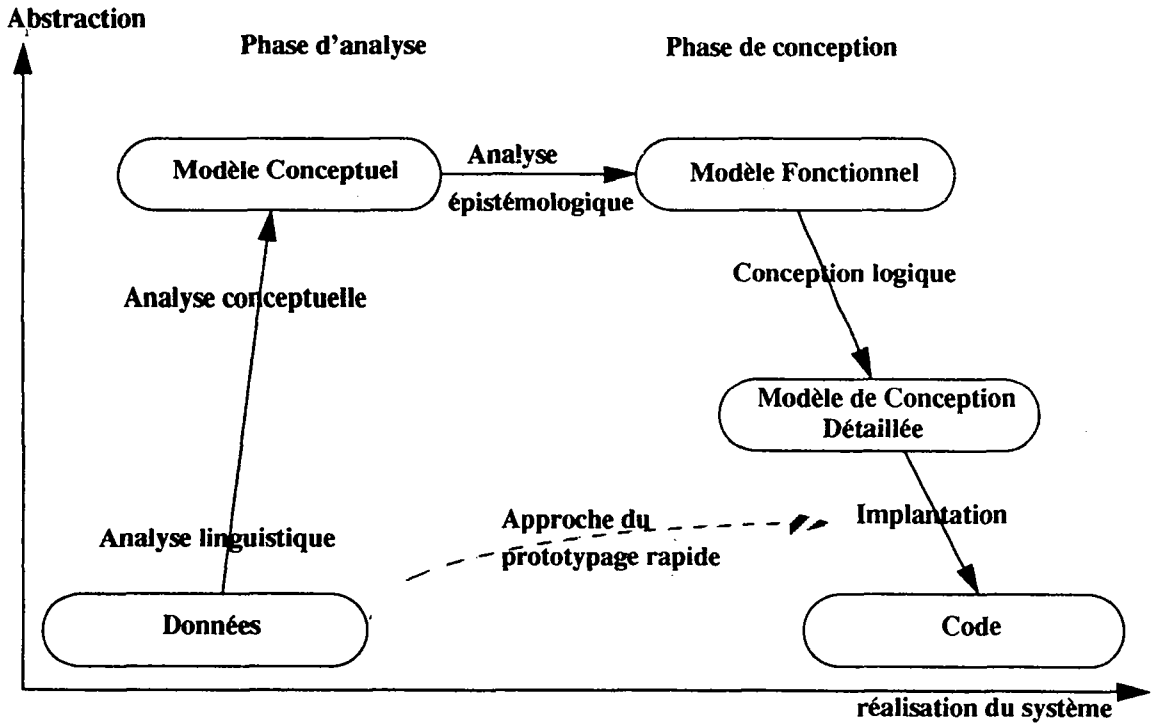


Figure: Modèles et espaces de développement de KADS (Extrait de [Schreiber 88])

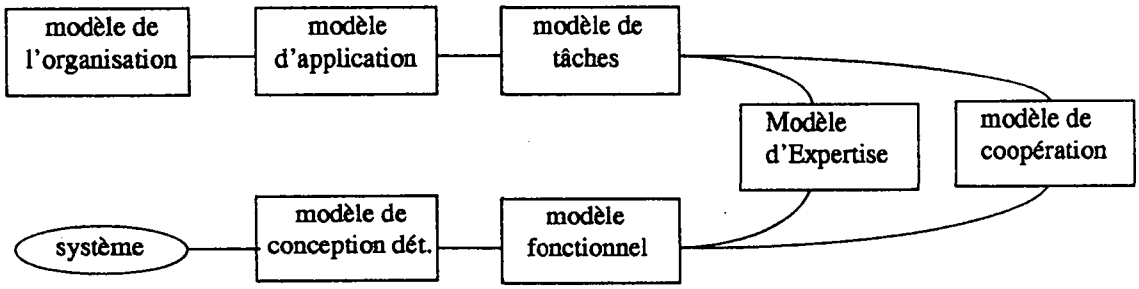
KADS repose sur l'hypothèse que les phases d'analyse et de conception doivent être séparées : il n'y a pas de retour sur la phase de conceptualisation une fois celle-ci achevée, mais le raffinement de modèles exige l'alternance de recueils et d'analyses de ceux-ci. Le cogniticien peut donc extraire les connaissances utiles de l'expert (pour l'application) indépendamment des conditions d'implantation et vision par vision (il existe plusieurs sous-modèles au Modèle Conceptuel comme

1. KADS (Knowledge Acquisition and Documentation Structuring) est issue d'un projet de recherche (fondé en partie par les programmes Esprit de la CEE) réunissant STC Technonogy Ltd, SCICON Ltd, KBSC SCS, Cap Sogeti Innovation et l'Université d'Amsterdam.

le montre la section suivante). L'information recueillie dans le Modèle Conceptuel est ainsi moins biaisée (cf chapitre précédent) est structurée et d'un haut d'abstraction. Les majuscules de Modèle Conceptuel seront omises lorsqu'il n'y aura pas d'ambiguité sur le modèle désigné.

Les modèles-étapes de KADS

Les différents modèles proposés par KADS permettent non seulement la "capture" et la représentation en machine d'un savoir-faire, mais aussi la compréhension du problème réel, de l'organisation et de ses besoins. Ce sont donc aussi des documents de référence (mémoire collective) [Wielinga 92].



- Le modèle de l'organisation décrit l'environnement du SBC (avant et après son introduction pour apprécier son impact) ainsi que les interactions (au niveau humain ou social) avec les utilisateurs. Doivent donc être spécifiés (entre autres) : l'objectif du manager, le rôle du ou des experts, les types d'utilisateurs, leurs besoins et utilisations du SBC. Ceci est important pour permettre de bien cibler l'utilité et la fonction de l'outil, d'assurer son adéquation aux besoins des utilisateurs, de s'assurer de sa faisabilité [Campbell 89].
- Le *modèle d'application* définit quel est le problème à résoudre, l'utilité d'un SBC, sa fonction et ses contraintes techniques (langage, système d'exploitation, performance, etc).
- Le *modèle de tâches* spécifie comment la fonction est décomposée par l'expert en tâches et sous-tâches, indique leurs entrées-sorties et les agents internes ou externes (l'utilisateur) qui les exécutera. Ceci est développé dans la section 2.1.
- Le *modèle de coopération* recense les tâches de transferts (e.g. «obtenir-de», «fournir-à» (l'utilisateur)) entre le système et son environnement (ex: l'utilisateur) et précise ainsi les rôles respectifs (qui fait quoi). Ce modèle est développé dans la section 2.2 ainsi qu'au chapitre 5, afin de l'apdapter pour prendre en compte les expertises liées aux explications.
- Le *Modèle d'Expertise* utilise le modèle de tâches et décrit le comportement et le type de connaissances nécessaires au système pour la résolution du problème, indépendamment de toutes contraintes d'implantation. Il s'agit donc d'un modèle de la connaissance de l'expert, subjectif car il s'agit d'une vision particulière du monde réel, mais ce n'est pas un modèle cognitif de l'expert car il est orienté par ce que peut et doit faire le futur SBC. Ce modèle joue donc un peu le rôle de spécifications fonctionnelles dans les développements classiques: c'est "un cadre sémantique partagé par des utilisateurs d'un programme et ses développeurs qui leur permet de communiquer" [Aussenac 89] ou encore "le modèle qu'a en tête le concepteur quand il va implanter le SBC" [Brunet 91]. La structure de ce Modèle d'Expertise (ou son langage), caractéristique de KADS, sera détaillée dans la section suivante.
- Le *Modèle Conceptuel* est la réunion des deux derniers modèles et décrit donc de façon abstraite les objets, comportements et opérations nécessaires au futur SBC. Par extension, il désigne toutes les activités menant à ce modèle (analyse linguistique puis conceptuelle). Certains générateurs de S.E. interprète partiellement le Modèle Conceptuel pour certains environnements ou type de tâches : Model-K [Karbach 91], ZDEST-2 [Tong 88].

- Une phase de conception plus "classique" et plus courte que celle de l'analyse (ex: 1 à 2 mois pour un prototype ayant demandé 6 mois d'analyse [Brunet 91]) comprenant :
  - le *modèle fonctionnel* qui correspond à l'architecture fonctionnelle du futur SBC et s'obtient par une analyse épistémologique du Modèle Conceptuel; là peuvent intervenir les termes du formalisme de représentation des connaissances du langage d'implantation;
  - le *modèle de conception détaillée*, après une analyse logique effectuée sur le modèle fonctionnel.

La spécification du Modèle Conceptuel est au «knowledge level» (cf [Newell 82] pour une longue description) mais comme nous l'avons vu, peut parfois être exécutable. Dans ce cas, la phase de conception a pour but principal de transformer cette spécification en un système efficace et utilisable ([Breuker 87]).

Ce cycle de vie peut être adapté (enrichit ou accéléré) en fonction du domaine de l'expertise ou d'autres contraintes (temps, etc.). Pour remplir ces modèles, KADS offre :
- un langage pour le Modèle Conceptuel (KCML; cf 2.1);
- une bibliothèque de tâches génériques, sorte de Modèles Conceptuels génériques à **combiner, spécialiser et remplir** en fonction du domaine de l'expertise (il n'est en effet pas aisé de construire un Modèle Conceptuel à partir de rien; les tâches génériques guident le cogniticien dans l'extraction et l'analyse de la connaissance de l'expert; ceci est développé au 2.4);
- un module de conception qui décrit comment le Modèle Conceptuel et les besoins externes peuvent être traduits dans une architecture adéquate (cf section 3);
- un module de modalités spécifiant la coopération et la communication du SBC avec l'utilisateur ou avec d'autres systèmes.

KADS permet l'utilisation de toutes les techniques de recueil des connaissances et propose l'outil SHELLEY ([Anjewierden 92]) pour analyser les données (éditeur de protocoles), construire un Modèle Conceptuel et les structures de la connaissance statique du domaine. Des techniques hypertextes y sont utilisées. D'autre outils existent également pour faciliter la conception.

2 La phase d'analyse

2.1 Le modèle de tâches

Pour créer le modèle de tâches, il faut d'abord identifier la tâche réelle globale contenant toutes les sous-tâches avec lesquelles les fonctions à automatiser ont des dépendances d'entrées ou de sorties, ce qui pose généralement peu de problèmes.
Cette tâche est **décomposée** en différentes sous-tâches dont les **interdépendances** sont identifiées (réseau de dépendance de données). Ces *transferts* d'objets (nous les appellerons des **ingrédients**) entre les sous-tâches conduiront à des spécifications sur leur **communication** (dans le modèle de coopération pour les **tâches de transferts** (sous entendu entre le système et l'utilisateur) et dans le Modèle d'Expertise pour les transferts internes au système. Lors de la conception, ces spécifications sont raffinées et prennent en compte le matériel ou les logiciels utilisés.

Les sous-tâches sont alors attribuées à des «**agents**» (systèmes ou type d'utilisateurs). Cette attribution peut se faire en parallèle avec la décomposition pour tenir compte des compétences des utilisateurs. Le résultat de cette phase est une ou plusieurs **distributions** (voire donc décompositions), plusieurs pour par exemple, tenir compte des types d'utilisateurs.

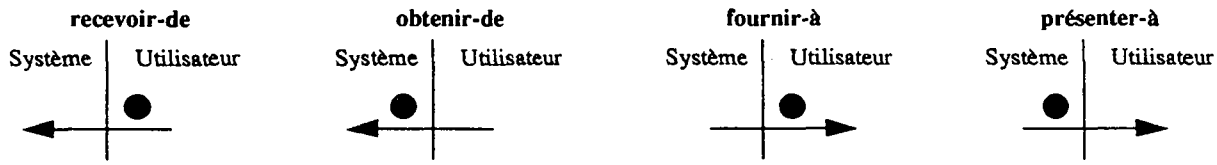
[De Greef 92] donne certains conseils pour la décomposition de la tâche globale en sous-tâches (arbre ET/OU, etc. cf p94) et pour la distribution : limitations des dépendances, exploitation des compétences, etc. Il préconise pour **tester** ces deux actions, l'utiliser de la **technique du Magicien d'Oz** (où l'expert joue le rôle du futur SBC en communiquant avec l'utilisateur via un terminal, mais toutefois en langage naturel). Il semble intéressant d'utiliser les relevés de ces expériences pour spécifier la **communication** associée à chaque tâche de transfert et les stratégies explicatives à sélectionner.
Notons cependant que la distribution des sous-tâches entre le système et l'utilisateur peut parfois ne pas être totale (i.e. maintien de tâches **partagées** mais où les rôles de chacun sont cependant clairs), et ce pour au mois trois raisons.

- Il ne peut être déterminé de manière fixe qui va faire quoi. L'allocation doit donc être dynamique, après négociation entre l'utilisateur et le système. Ce dernier doit alors avoir une liste d'alternatives pour choisir.
- La sous-tâche peut devoir être effectuée en parallèle par le système et l'utilisateur, par exemple à des fins d'enseignement.
- Dans la sous-tâche, l'un des deux agents donne à l'autre des instructions et ce dernier les exécute. C'est par exemple le cas d'un Système de Gestion de Fichiers qui exécute des commandes ou d'un débutant dans l'art culinaire qui suivrait, pour la préparation d'un diner, les consignes dictées par un SE.

Notons que les divers cas de coopération entre le système et l'utilisateur que nous avons vu au début du chapitre précédent, sont pris en compte.
Le modèle de tâche ainsi formé sert d'entrée à l'analyse de l'expertise d'une part, et est raffiné pour inclure les aspects contrôle et communication d'autre part. Il s'agit là de la **seconde phase de l'analyse de la coopération** qui consiste tout d'abord à spécifier pour chaque tâche de transfert qui de l'utilisateur ou du système a l'**initiative** de la communication.
Puis, pour chacune de ces tâches de transfert, ou pour l'ensemble, il faut spécifier **les modèles de communication : «présentation» (ex: explications) et «dialogue»**. Enfin, il faut assurer la **synchronisation** entre ces tâches et le Modèle d'Expertise. Quoique la place de ces deux dernières importantes étapes soit très ambiguë dans [De Greef 92], il sera ici logiquement considérée qu'elles font partie de la construction du modèle de coopération (**en phase finale**).

2.2 Le modèle de coopération

Le *modèle de coopération* n'est dans le KADS standard qu'une simple annexe du modèle de tâches, définissant sommairement les tâches de transfert : pour chacune, il précise qui, de l'utilisateur ou du système, a l'initiative de l'échange, et quel est le type d'«ingrédients» transférés : «information», «connaissance», «talent», etc. [De Greef 92].
Si l'utilisateur a l'initiative, i.e. débute la communication, cela signifie qu'il peut interrompre le processus de résolution. Sinon, c'est le système qui rentre en mode de communication. Suivant l'origine (et donc le contrôle) du flot de données, on obtient quatre types de tâches de transferts.



Ceci exprime le point de vue du système. La flèche symbolise le flot des ingrédients et le point, l'initiative.

Un cinquième type de tâche de transfert, «négocier», peut être utile pour transférer des informations sur (le type de) la coopération ou sur la résolution de problème. Exemples : sélection d'une distribution de tâches, correction d'une information donnée préalablement, etc.

Trois types majeurs d'ingrédients peuvent être distingués : «l'information», la «connaissance» et le «talent».

- L'**information** réfère à des états spécifiques du monde réel ou conceptuel (il y a donc beaucoup d'analogie avec les méta-classes des structures d'inférences). Typiquement :
  - des données, comme une valeur de variables ou une description de structure au sens KADS;
  - des états de problèmes
  - des états internes ou mentaux : intention, évaluation, histoire ou état actuel de la résolution, processus de communication, etc. ; on retrouve particulièrement des méta-classes de même nom en entrées ou sorties des structures d'inférences.
- La **connaissance** n'est pas spécifique d'une situation mais générique. Elle peut être utilisée en tant qu'ingrédient pour des buts d'explication ou d'enseignement.
- Le **talent** est transféré quand l'objectif est d'instruire l'autre agent sur la façon d'exécuter une sous-tâche (idem pour la connaissance). Ceci arrive par exemple lorsque les compétences de l'utilisateur sont moins grandes que prévues.

D'autres types d'ingrédients peuvent être nécessaires pour :
- produire des explications qui ne contribuent pas à la coopération, si ce n'est indirectement en augmentant la confiance que l'utilisateur a dans le système
- transférer des informations sur le processus de communication lui-même, par exemple lorsqu'un des deux agents ne comprend pas l'autre ou veut changer le sujet de la communication.

L'alternance entre les tâches de résolution de problèmes et les tâches de transfert, c'est à dire finalement la coopération ou répartition du travail entre le système et l'utilisateur, est définie par des structure de tâche, donc statiquement. Nous relevons au chapitre 5, les inconvénients de cette définition statique. D'autant que la ou les structures de tâches sont ensuite insérées aux niveaux tâches et parfois statégie du Modèle d'Expertise. Des connaissances de répartition du travail sont ainsi mélées avec des connaissances de résolution de problèmes. Voici un exemple typique de structure de tâche donnée par [De Greef 92] et mettant en valeur les tâches de transferts :

```
process (problème, solution)
  négocier (type-d-utilisateur, structure-de-tâche)

structure-de-tâche-1
  résoudre (problème, solution)
    recevoir-de (problème)
   monitor (interruption-de-l'utilisateur-1) => fournir (trace-de-résolution-de-problème) {
      spécifier (modèle-du-système)
      ...                                          //grâce à la construction monitor,
      obtenir-de (données)                         //l'utilisateur peut interrompre ce bloc
      fournir-à (but-des-données)
      monitor (interruption-de-l'utilisateur-1) => négocier (données) {
        ...
      }
      ...
    }
    présenter-à (solution)    //suite page suivante
    présenter-à (solution)    //reprise
    fournir-à (justification-de-la-méthode-de-résolution-de-problème) { //l'utilisateur
       classifier (problème)                                            //a le contrôle
       abstraire (caractéristiques-du-problème, type-du-problème)
       spécifier ...
     ...
```

2.3 Le langage de modélisation de l'expertise

KCML permet une représentation intermédiaire des données de l'expertise préalablement (mais indépendamment) à la conception/implantation du SBC. Il structure ces données en quatre niveaux relativement indépendants au lieu de deux (domaine et contrôle) dans les méthodologies traditionnelles. La sous-section 2.2 illustre chacun de ces niveaux sur un exemple de diagnostic dans un système audio [Schreiber 91].

- Le niveau **domaine** décrit la connaissance statique du domaine (et ses principes), indépendamment de son utilisation et peut donc être utilisée (si elle est complète) aussi bien pour la résolution de problèmes que pour les explications, l'enseignement, etc. (tout comme l'expert peut utiliser son savoir). Il rassemble les ***concepts*** du domaine (savoirs génériques ou instances décrits chacun par un nom et des propriétés avec leurs valeurs possibles), les ***relations*** entre concepts (variantes des liens de spécialisation, de composition, d'appartenance, de relations spatiales, etc.) et les relations entre propriétés de concept (liens de causalité, d'association ou de relations fonctionnelles, liens temporels, etc.). Suivant le domaine, d'autres représentations peuvent être utilisées ou ajoutées : formules mathématiques, frames, langage de règles, etc.
  Quelques exemples : les concepts quantifiables contiennent des propriétés comme la quantité, la dimension, ... avec des contraintes dessus; les concepts de diagnostic peuvent être des évidences, des hypothèses, des causes, des actions tandis que ceux des tâches de conception peuvent être des plans, des spécifications, des structures, des contraintes, etc.
  Dans le domaine médical, les concepts peuvent être les divers désordres, les causes de ces désordres, les symptômes qu'ils entraînent, les données du laboratoire, ... et pour les relations, les liens hiérarchiques entre les désordres, les liens causaux entre causes et symptômes ou entre causes et désordres, etc.
  Ce niveau peut être vue comme une théorie déclarative du domaine qui, exploitée par un simple mécanisme de déduction, serait suffisante (en théorie) pour résoudre tous les problèmes résolvables par la théorie. En pratique, ce ne serait pas le cas, compte-tenu des limitations des techniques actuelles de démonstrations de théorème. La résolution ne serait pas dirigée, partirait donc dans toutes les directions et déduirait de nombreuses connaissances inutiles (d'où la nécessité du niveau suivant).

- Le niveau **inférence** mentionne les inférences (dérivation d'information depuis d'autres informations) que l'on désire faire sur le domaine et pourquoi (mais ne précise pas quand ni comment les utiliser). Il distingue pour cela :
  - les ***«méta-classes» ou «rôles»*** qui sont des pointeurs de concepts du domaine utilisés au cours du raisonnement et dont le nom souligne justement leur rôle au cours du raisonnement; exemples : *hypothèse* et *solution* pour une infection, *symptôme* (déclencheur d'hypothèses) et *manifestation* (vérification d'hypothèses) pour de la fièvre (une typologie de ces méta-classes est donnée au 2.5);
  - les ***sources de connaissances*** (SC) qui décrivent comment une relation du domaine peut être utilisée pour réaliser des actions (décomposer, assembler, classifier, comparer, etc; une typologie est aussi donnée au 2.5); chaque SC une inférence élémentaire (indivisible) : son exécution n'est pas controlée par d'autres parties du modèle (le contrôle explicite se trouvent aux niveaux suivants) et elle réussit ou échoue (et peut donc être vue comme une simple procédure sur le domaine).
  Les SC ont des méta-classes pour arguments d'entrée et de sortie, référençant ainsi les concepts du domaine sur lesquels elles s'appliquent. Pour exprimer ceci, ainsi que le contrôle, plusieurs formalismes de représentations sont possibles. Voici donc quelques exemples de SC

d'abord en prolog puis, de façon plus claire et plus complète, en langage structuré.
Les (représentants des) méta-classes sont en italiques dans les deux cas.

```
//C1 et C2 étant des concepts, «père-de» une relation
raffiner(C1,C2):- hypothèse(C2),hypothèse(C1), père-de(C2,C1). //par spécialisation
généraliser(C1,C2):- hypothèse(C1),hypothèse(C2), père-de(C1,C2).

//différencier entre deux hypothèses (C0 étant aussi un concept et «attribut» une relation)
différencier(C1,C2,C3,A):- hypothèse(C1), hypothèse(C2),
                           père-de(C0,C1), père-de(C0,C2), attribut(C0,A),
                           valeur(A,C1,V1),valeur(A,C2,V2), différent(V1,V2).


//composant étant un concept et «composant-de» une relation, «->» pour «réfère à»
SC une-décomposition(in modèle-du-système -> composant;     //«in» devant une méta-classe
                     out hypothèse -> composant)     //d'entrée; «out» devant une de sortie
       décomposer(modèle-du-système,hypothèse) -> composant-de(composant,composant).

//explications: pour effectuer l'action de «décomposer» un modèle du système (sur lequel
 porte le raisonnement, le problème) afin de générer diverses hypothèses (dans le but de
 trouver l'origine d'une anomalie par exemple), cette SC utilise la relation
 «composant-de». Elle aurait pu mentionner plusieurs relations (en coopération ou en
 concurrence) et indiquer le contrôle ou au moins la méthode à employer (mais ceci peut
 être omis lorsque la méthode est évidente, ici : la descente dans la hiérarchie de la
 relation; notons que «évidente» ne veut pas dire que les techniques de mises en oeuvre
 soient simples). Elle spécifie explicitement les concepts sur lesquels elle travaille.
 Dans le cas de l'exemple de la section suivante, les instances de «composant» sont le
 système audio et ses sous-parties (amplificateur, etc.).
```

Ce dernier formalisme montre bien le vrai rôle d'une SC (comme d'une méta-classe) : une «poignée» pour le raisonnement sur le niveau domaine (les niveaux suivants n'utilisent pas directement les concepts et les relations). Les objets et les relations, étant seulement référés par des méta-classes, peuvent être utilisés plusieurs fois, sous différents points de vue (appelations), sans redondance d'information (une même méta-classe pouvant bien sûr être utilisée par plusieurs SC).
La combinaison des SC via les méta-classes, pour accomplir une tâche de résolution de problèmes peut être représenté graphiquement par des ***structures d'inférences***. C'est un réseau statique montrant quelles inférences peuvent être faites pour atteindre un but, mais pas l'ordre de l'exécution ou la stratégie utilisée.

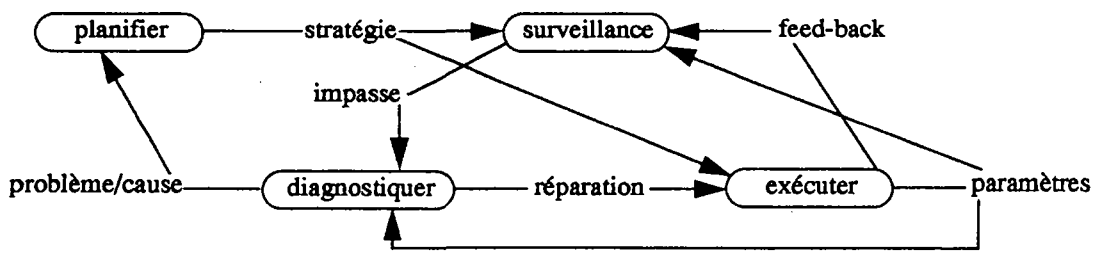
- Le niveau **tâche** explicite comment contrôler les inférences élémentaires pour atteindre un but. Un but est un état espéré (concept avec des attributs spécifiques qui peut être décrit par une méta-classe i.e. un état du processus de résolution) souvent accompagné du nom de l'action nécessaire pour l'obtenir (e.g. le nom de l'action effectuée par une SC). Il y a généralement plusieurs tâches par but (plusieurs méthodes) et quelquefois plusieurs buts par tâche.
  Les tâches peuvent se décomposer en sous-tâches. Il y a trois types de (sous-)tâches :
  - ***primitive*** (une inférence élémentaire i.e. une SC spécifiée au niveau inférence plus éventuellement un contrôle sur l'appel);
  - ***composée*** (comme la tâche qui la contient; la récursivité est ainsi possible);
  - ***de transfert*** i.e. exigeant une interaction avec un agent externe, généralement l'utilisateur; elles sont spécifiées dans le modèle de coopération; quatre types peuvent être distingués : obtenir-de, recevoir-de, présenter-à et fournir-à (à l'agent extérieur; précisions chapitre 5).

Le moyen d'atteindre un but est décrit par un langage de règles ou par de l'algorithmie classique (cf section suivante) et/ou par le nom de la méthode utilisée (ex: association heuristique, chainage arrière, etc.). La décomposition des buts en sous-buts (ou contrôle de la tâche sur les sous-buts) est ainsi visible dans les ***structures de tâche***. Ce sont des stratégies fixées soit par le concepteur, soit quelquefois par le niveau stratégie (cf ci-dessous).
Quelque soit le moyen de contrôle utilisé, les tests s'effectuent sur des méta-classes ou sur des ***termes de contrôles*** i.e les définitions d'un ensemble de méta-classes (ex: le «différentiel» est l'ensemble des «hypothèses» actives pour une tâche de diagnostic).
Les tâches ne réfèrent qu'au niveau inférence. Quoique difficile, cette séparation domaine, inférences, mécanismes de résolution est souvent utile : clarté, généricité, non redondance, ...

- Le niveau **stratégie** détermine quels buts doivent être poursuivis pour résoudre un problème (le comment étant l'affaire du niveau tâche). Il permet aussi de réagir en cas d'échec (de toutes les méthodes) ou d'impasse (information non disponible ou contradictoire). Dans de tels cas, il doit suggérer de nouvelles approches ou moyens d'acquérir de l'information en faisant des hypothèses [Jansweijer 88]. Grâce à ce niveau, le système doit savoir s'il peut ou non résoudre le problème qu'on lui pose (une solution fausse coûte plus chère que l'absence de solution).
  Nous avons vu que les décompositions en sous-tâches spécifiées dans les structures de tâche représentent une stratégie fixée par avance. De façon à pouvoir s'adapter à des problèmes différents ou éviter de refaire les mêmes erreurs, une solution est d'équiper de modèle d'un planificateur qui ***génèrerait*** dynamiquement un arbre des (sous-)buts (et donc quelques structures de tâches) par des méthodes générales de résolution de problèmes (analyse par les buts, classification heuristique, etc.) et adaptation à l'environnement de la tâche et à la modalité du système (façon dont le système doit communiquer avec l'utilisateur). Ceci est indispensable pour les SBC devant être ou capable d'un apprentissage intelligent [Breuker 87]. Voici un exemple de méta-règle pouvant faire partie du planificateur :

```
SI le but en cours est de trouver un solution ET que la modalité est la consultation ET
   s'il y a des SC reliants des conclusions aux hypothèses ET
   s'il y a des SC permettant de raffiner les hypothèses
ALORS décomposer le but en cours en : - rassembler les conclusions
                                      - générer (d'après celles-ci) des hypothèses
                                      - raffiner ces hypothèses.
```

En général, la planification n'est pas une solution suffisante pour obtenir un comportement de résolution flexible. Quand un plan est exécuté, sa progression doit être surveillée et les échecs possibles surmontés (réparés). Voici une structure pour un raisonnement stratégique :



Quasiment jamais utilisé dans les SBC développé avec la méthodologie KADS, ce niveau reste essentiellement un domaine de recherche (ex: actuel projet Esprit REFLECT) et est actuellement repensé dans le projet KADS-II [KADS-II 92Lib].

Ainsi, tandis que les deux premiers niveaux représente la partie déclarative des connaissances extraites, les deux derniers niveaux définissent le contrôle (éventuellement dynamique) sur celles-ci.

Le caractère informel de ces spécifications leur donnent une certaine clarté, ce qui a contribué au succès de KADS. Le revers de la médaille, c'est que le Modèle Conceptuel n'est pas opérationnel et peut difficilement être testé. Or ceci est indispensable pour une application complexe car il est impossible de concevoir toutes les interactions possibles entre tous les composants du modèle [Voss 90]. L'utilisation conjointe de modèles formels et informels est un des principaux sujets de recherche du projet KADS-II. Une solution consisterait alors à pouvoir extraire des prototypes. Notons cependant que des outils permettent déjà d'interpréter partiellement des Modèles Conceptuels mais en n'offrant qu'une palette limitée de formalismes de représentation et de mécanismes d'inférence ou de contrôle (chaînage arrière, blackboard, etc.) donc rarement adaptée pour implanter correctement toute l'expertise.

Afin de préciser un peu toutes ces notions, revoyons les sur un exemple.

2.4 Exemple de diagnostic d'un système audio

Niveau domaine

Il contient au moins deux **concepts** «composant» et «test» dont les instances sont respectivement les éléments du système audio et les tests effectués pour connaître l'état de ce système.
Chacun de ces concepts a une ou des **propriété(s)**. Ex: «test:valeur» (pour valeur du test).
Quatre **relations** : composant EST-UN composant (hiérarchie des sous-types de composant), composant SOUS-COMPOSANT-DE composant (hiérarchie des sous-parties de composant), composant:valeur-état CAUSE composant:valeur-état (relations causales entre états normaux), test:valeur INDIQUE composant:état-valeur (tel résultat de test indique tel état).

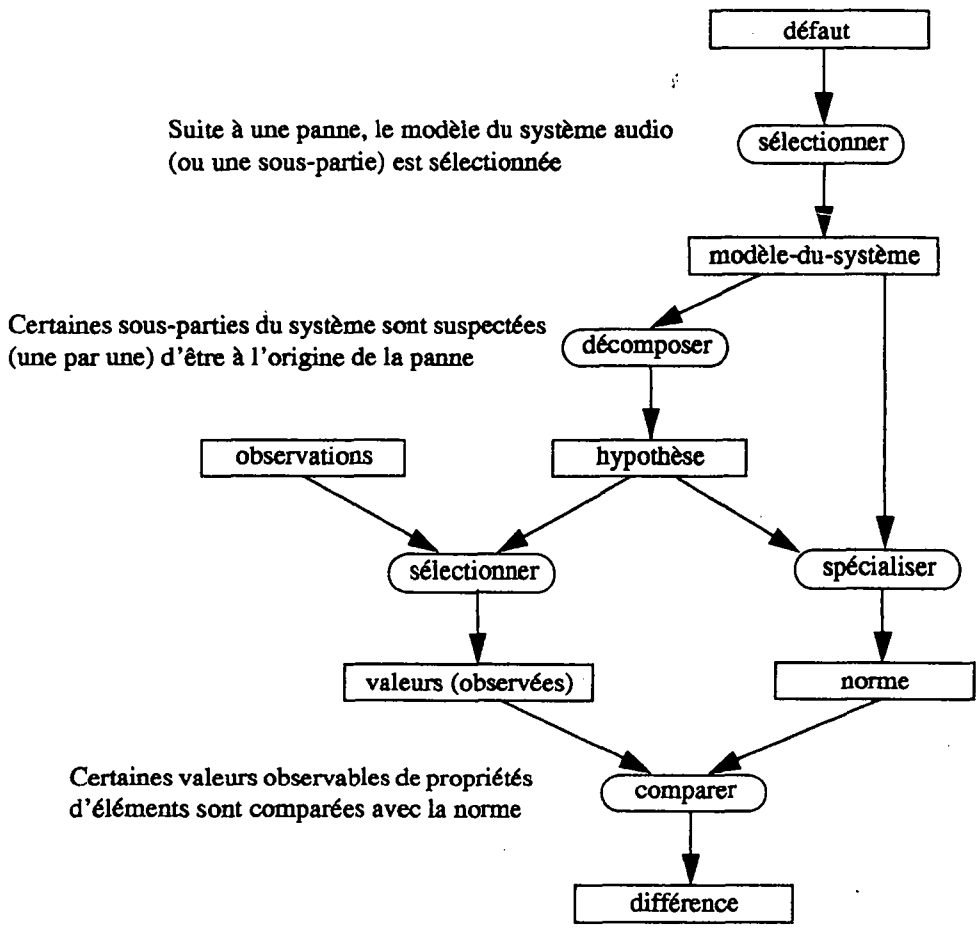
Ce ***schéma du domaine*** minimal étant décrit, le ***domaine*** lui-même peut l'être aussi :
- système-audio, platine-à-cassette, CD, système-haut-parleur, etc. sont des composants
- système-audio a pour sous-composant système-haut-parleur, amplificateur et platine-à-cassette, CD, ..., système-haut-parleur a pour sous-composant ..., etc
- propriété des composants: amplificateur:signal-d-entrée (platine,tuner,CD), amplificateur:puissance (on,off), ... et des tests: bouton-alimentation: (on,off), ...
- relations: amplificateur.signal-d-entrée=CD et amplificateur:sélection-d-entrée=CD CAUSE amplificateur:signal-de-sortie=CD, sélecteur-d-entrée=X INDIQUE amplificateur.signal-d-entrée=X, ...

Le schéma du domaine aurait pu inclure les notions de «faute» ou de «rectitude», préciser les cardinalités des relations, etc. Il peut être utilisé dans d'autres systèmes de diagnostic.
Des *primitives épistémologiques* pour décrire la théorie du domaine (structure, concept, propriété, relation entre propriétés, relation entre structures) ont été proposées par Brachman et Schmolze ([Brachman 85]). Une ***structure*** est un ensemble de concepts liés par des relations (souvent du type «a-pour-attribut»). On utilise une structure pour représenter les objets complexes (ex: le système-audio aurait pu être vu comme une structure). Des ensembles d'instances peuvent également être utiles pour du raisonnement quantitatif ou probabilistique.
Le concepteur peut en fait, choisir les primitives qu'il désire (e.g. de bases de données) du moment que le formalisme résultant est complet et consistant. Ces types de problèmes de représentation de la connaissance fait partie du domaine des recherches en épistémologie ou ontologie. Pour plus de détails sur ces dernières, lire [Skuce 90] [Lenat 90] [Breuker 87, pp 32-35].

Niveau inférence

```
SC une-décomposition(in modèle-du-système -> composant;  out hypothèse -> composant)
      décomposer(modèle-du-système, hypothèse) -> sous-composant-de(composant, composant).
```

Cet exemple de SC montre que grâce aux méta-classes, les SC réfèrent les concepts du domaine et non directement ses éléments. Ceux-ci sont donc utilisables à plusieurs fins (résolution, connaissances profondes pour les explications, etc.) et peuvent avoir autant de noms que d'utilisations. Même s'il y a souvent interaction entre la conceptualisation du domaine et la spécification des inférences du processus de résolution (la structure du domaine devant s'adapter aux spécifications de l'ensemble des inférences), il est utile de les documenter (au moins) séparément.
Les différentes étapes formées par les inférences élémentaires lors du processus de résolution sont visibles dans une structure d'inférence. Conventionnellement, une méta-classe est entourée par un rectangle tandis qu'une SC est représentée par le nom de l'action qu'elle effectue entourée par un ovale Voici une pour notre exemple (cette structure d'inférence est celle de «la tâche générique de diagnostic systématique» et est donc détaillée au 1.1 de l'annexe).



Les SC et méta-classes ici utilisées sont inspirées de la typologie présentée au 2.5 (qui n'est qu'un conseil, le concepteur choisissant les abstractions qu'il désire).

Niveau tâche

Voici, avec les termes définis dans la présentation du niveau inférence, les tâches de notre exemple. Les SC sont en italique et des flèches séparent arguments d'entrée et arguments de sortie des sous-tâches. Notons que les tests et les arguments des tâches sont soit des méta-classes soit des termes de contrôles (ensemble de méta-classes). Un modèle de cette tâche de diagnostic est présentée en annexe.

```
tâche diagnostic-systématique
  but : trouver le plus petit composant ayant un comportement incorrect s'il y en a un
  termes de contrôle
     différentiel = ensemble des hypothèses en cours
     sous-système-incohérent = sous-partie du système ayant un comportement incohérent
  structure de tâche  diagnostiquer-systématiquement (réclamation -> sous-système-incohérent) {
     sélectionner (faute -> modèle-du-système)
     générer-hypothèses (modèle-du-système -> différentiel)
     REPETER
           tester-hypothèses (modèle-du-système,différentiel  -> sous-système-incohérent)
           générer-hypothèses (sous-système-incohérent -> différentiel)
     TANT QUE différentiel non vide
  }

tâche générer-hypothèse
  but : générer un nouvel ensemble d'hypothèses grâce à la décomposition
  termes de contrôle
  structure de tâche   générer-hypothèse (modèle-du-système -> différentiel) {
     décomposer (modèle-du-système -> différentiel)
  }

tâche tester-hypothèse
  but : chercher une hypothèse du différentiel se comportant anormalement
  termes de contrôle
  structure de tâche   tester (modèle-du-système,différentiel  -> hypothèse) {
     POUR chaque hypothèse du différentiel
           spécialiser (modèle-du-système,hypothèse -> norme)
           sélectionner (hypothèse -> observations)
           obtenir (observations -> valeurs)  /* tâche de transfert avec l'utilisateur */
           comparer (norme,valeurs  -> différence)
     TANT QUE différence = faux
  }
```

Cet exemple n'a pas de niveau stratégique.

Utile pour comprendre le processus de raisonnement au niveau d'abstraction de l'expert (et dans son langage), ce modèle à 4 niveaux se rapproche d'autres modèles existants, à certaines différences près pour le placement de quelques types de connaissances. Cependant, la plupart n'ont qu'un niveau tâche et un niveau domaine, ce dernier étant dédié à des méthodes de résolution.

2.5 Typologies et réutilisabilité

Une typologie des ST (c'est ainsi que nous appelerons les modèles des SC, i.e. les actions Typiques qu'elles effectuent et sur quels types d'arguments) a été proposée par [Breuker 87] et a fournit le support de construction d'un nombre considérable de modèles (c'est une aide pour l'analyse des données).
Elle est basée sur les opérations possibles avec les primitives épistémologiques définies en KL-ONE [Brachman 85]. Cet ensemble de primitives comprend : concept, instance de concept, attribut de concept, valeur d'attribut et structure de concept (laquelle est à son tour un concept). Un concept peut être modifié (valeur d'attribut) ou servir à générer un concept différent (éventuellement grâce à d'autres concepts). D'où la proposition suivante [Breuker 89] (qui est développée pour les explications au chapitre 6) :

génération de concept (de lien si le concept existe déjà) **ou d'instance**
instancier (concept -> instance) //création d'une instance
identifier (instance -> concept) //(ou classifier) inverse de instancier (ex: Fido est un chien)
généraliser (ensemble d'instances -> concept // [1]
abstraire (concept -> concept) //le 2ème concept a moins d'attributs (ex: lien «est-un»)
spécialiser (concept -> concept) // ou raffiner (et donc spécifier), inverse d'abstraire
sélectionner (ensemble de concept -> concept) //c'est une forme dégradée de spécialiser, un concept spécifique est sélectionné (très utilisé)

comparaison
comparer (valeur de X, valeur de Y -> diff) //les valeurs des concepts X et Y en entrée sont comparées et diff exprime une valeur ou un concept de (non) égalité)
matcher (structure X, structure Y -> concept-exprimant-la-différence-de-structure) //c'est une comparaison de structure, par exemple par unification

modification de concept (modification de valeurs)
assigner_valeur (attribut de concept -> attribut valué) //ex: valeurs par défaut
évaluer (structure X -> concept de X valué) //la valeur dépend des relations entre les concepts (et/ou leurs instances); ex de structure: formule ou contrainte

modification de structure (les plus fréquentes et de façon assez générique car il y en a, en fait, autant que de types de structure)
assembler (ensemble d'instances -> structure «partie-de»)
décomposer (structure «partie-de» -> ensemble d'instances) //inverse de assembler
transformer (structure X -> structure Y) //tri ou structuration des éléments de X

Cet ensemble est bien sûr incomplet. D'autres opérateurs tels que joindre, unir ou fusionner aurait pu être ajoutés. Les méthodes utilisées pour les actions sur les structures sont complexes et consistent en réalité en de nombreuses SC plus primitives (ex: assembler, SC très importante pour les tâches de synthèse, supportée par des méthodes de satisfaction de contraintes).
De plus, ces actions sont très générales donc peu adaptées à un domaine particulier et souvent ambiguës pour le cogniticien qui veut reprendre un ensemble d'inférences d'un Modèle Conceptuel existant. C'est un des sujets de recherche du projet KADS-II que de trouver des taxonomies spécifiques d'un domaine particulier (ex: diagnostic technique).

---

1. D'après les caractéristiques communes des instances, on cherche un concept réunissant ces caractéristiques (comme pour «**identifier**») ou l'on en développe un (c'est alors de l'**induction** et de nombreuses méthodes ont été développées pour cela, notamment en apprentissage [Charniak 85]). Dans la littérature, le terme de généralisation regroupe à la fois «généraliser» et «abstraire» et celui d'abstraction, à la fois «abstraire» et «spécialiser» (où il recouvre donc de nombreux types d'inférences liants un objet à un autre). De même, le terme de classification (en temps que méthode d'inférence) a une signification plus large que «identifier» (dans KADS, cette méthode est aussi appelée «classifier», bien que ce soit une source de confusion possible avec la tâche de classification). Ainsi, dans la littérature, classifier peut regrouper non seulement la décision de considérer que «Fido est un chien» (**«identifier»**) mais aussi celle de considérer que «un chien est un mammifère, un animal domestique, un compagnon» (**«abstraire»**). Cette confusion vient essentiellement de la non distinction entre l'utilisation des instances et des descriptions (concepts).

Il est moins aisé d'effectuer une typologie des méta-classes, pointeurs de concepts (i.e. poignées pour éléments du domaine) qui permettent, grâce à leur nom, d'abstraire l'intérêt des données à chaque étape du processus de résolution de problème. Nous verrons qu'elles sont très utiles pour les explications (chapitre suivant). Il y a de nombreuses façons d'interpréter le rôle des données lors de cette résolution et le langage courant contient relativement peu de termes précis à ce sujet. Voici donc une typologie provisoire, guidée par une vision «pseudo temporelle/causale» [Breuker 89] :

problème
  question
intention
donnée
  structure_de_donnée
    description-de-choix
    description-de-système
  donnée_individuelle
    contrainte
    variable (ou abstraction de donnée)
    symptome
      défaut (ou anomalie)
rôle intermédiaire de donnée de problème
  paramètre
  facteur
  résultat
    témoin (ou preuve)
rôle intermédiaire de donnée du domaine
  modèle-du-système
  hypothèse
  norme
  terme
solution
  diagnostic
  classe-de-décision
  plan
  conception
  modèle

Le Modèle d'Expertise peut être utilisé de plusieurs façons pour aider à supporter le processus d'acquisition des connaissances. L'une des plus puissantes est la ré-utilisabilité de ses éléments. Ainsi, puisque chaque niveau est relativement indépendant des autres, il peut servir à alimenter d'autres Modèles Conceptuels. Il est courant de reprendre soit la partie contrôle d'une autre application similaire (cf section suivante) ou son domaine. L'existence de bibliothèques de modèles évite de réinventer le feu et constitue selon [Wielinga 92], une condition indispensable pour améliorer l'état de l'art de l'acquisition des connaissances.

A un niveau plus fin (inférence), on peut citer deux axes de recherches pour la réutilisabilité :
- les typologies de SC et de méta-classes;
- les blocs d'inférences (ex: les tâches de diagnostic et de surveillance partagent un ensemble d'inférences, la comparaison entre la valeur attendue et la valeur observée) à assembler pour construire le Modèle Conceptuel. Cette vision est issue de travaux de Patil (1988) qui a montré comment l'on pouvait raffiner progressivement une tâche simple de diagnostic comme «générer et tester». C'est une vision plus dynamique de la construction de modèles que celle offerte par les tâches génériques (ceci est repris en dernière section de ce chapitre).

2.6 Modèles d'interprétation

Pour aider le cogniticien dans la construction d'un Modèle Conceptuel, KADS offre une bibliothèque de modèles d'interprétation i.e des **descriptions** de la connaissance requise pour effectuer une **classe particulière de tâches** aux niveaux inférence, tâches et stratégie. La nature du niveau **domaine** n'est généralement **pas spécifié**. N'importe quelle tâche réelle peut avoir son modèle d'interprétation correspondant (essentiellement en otant les références au domaine du Modèle Conceptuel de la tâche).

Il y a deux sortes de modèles d'interprétation : les modèles génériques (ou **tâches génériques** puisque ces modèles sont spécifiques d'une classe de tâches) et les **modèles** (ou tâches) **réels** (de la vie réelle). Les seconds sont des combinaisons des premiers : ce sont donc soit des Modèles Conceptuels réels en cours de construction, soit des (parties de) Modèles Conceptuels réels dont les références à un domaine spécifique ont été abstraites (ex: «évaluation de requêtes de clients») ou relativement abstraites (ex: «évaluation de prêts commerciaux»). Lors de la conception de KADS, les principaux efforts se sont donc portés sur les briques de bases : les tâches génériques (c'est pourquoi, en annexe, seuls ces modèles génériques sont détaillés).

Ce sont des tâches de résolution de problème minimales dans le sens où, étant donné un problème en entrée, elles produisent une «solution» (e.g. les ST n'apportent pas de «réponse» à leurs entrées). Ci-dessous est donnée une classification de ces tâches, par type de problème [Breuker 89]). Une **première distinction** est faite suivant que :
- la *structure interne du système* en cours d'investigation (ex: moteur, logiciel, patient, etc.; nous parlerons désormais simplement du «*système*») est *connue* (donnée, etc.) et des propriétés supplémentaires doivent être identifiées (**analyse**)
- ou bien que le système doit être construit et est le résultat du processus de résolution (**synthèse**).

Plus précisément, les tâches d'**analyse** ont pour ultime but d'établir les **propriétés** inconnues (ex: composant défectueux) d'un système ou son **comportement** (ex: prédiction d'un état). Elles laissent sa structure invariante.

Inversement, les tâches de **synthèse** ont pour but de trouver une **description structurelle** d'un système en termes d'un ensemble donné d'éléments (vocabulaire), de formalismes ou de structures partielles.
Les **spécifications** (descriptions du problème) devant être souvent identifiées en termes des-dits formalismes ou éléments, la synthèse peut initialement avoir un parfum d'analyse. Autre source de dégradation : plus le niveau des éléments/ structures partielles est élevé, plus la construction devient de la sélection de système, donc de l'analyse. C'est pourquoi, en fin de synthèse, lorsque le domaine a été décrit et que l'espace de solution est complètement déterminé, la tâche tend à devenir une tâche d'analyse.

Autre transition, les tâches qui **modifient** le système pour atteindre une fonctionnalité désirée sans affecter son intégrité (structurelle). De fait, il s'agit souvent de restaurer une fonctionnalité perdue. Voici donc cette typologie (les pages référées indiquent où ces tâches sont présentées dans l'annexe).

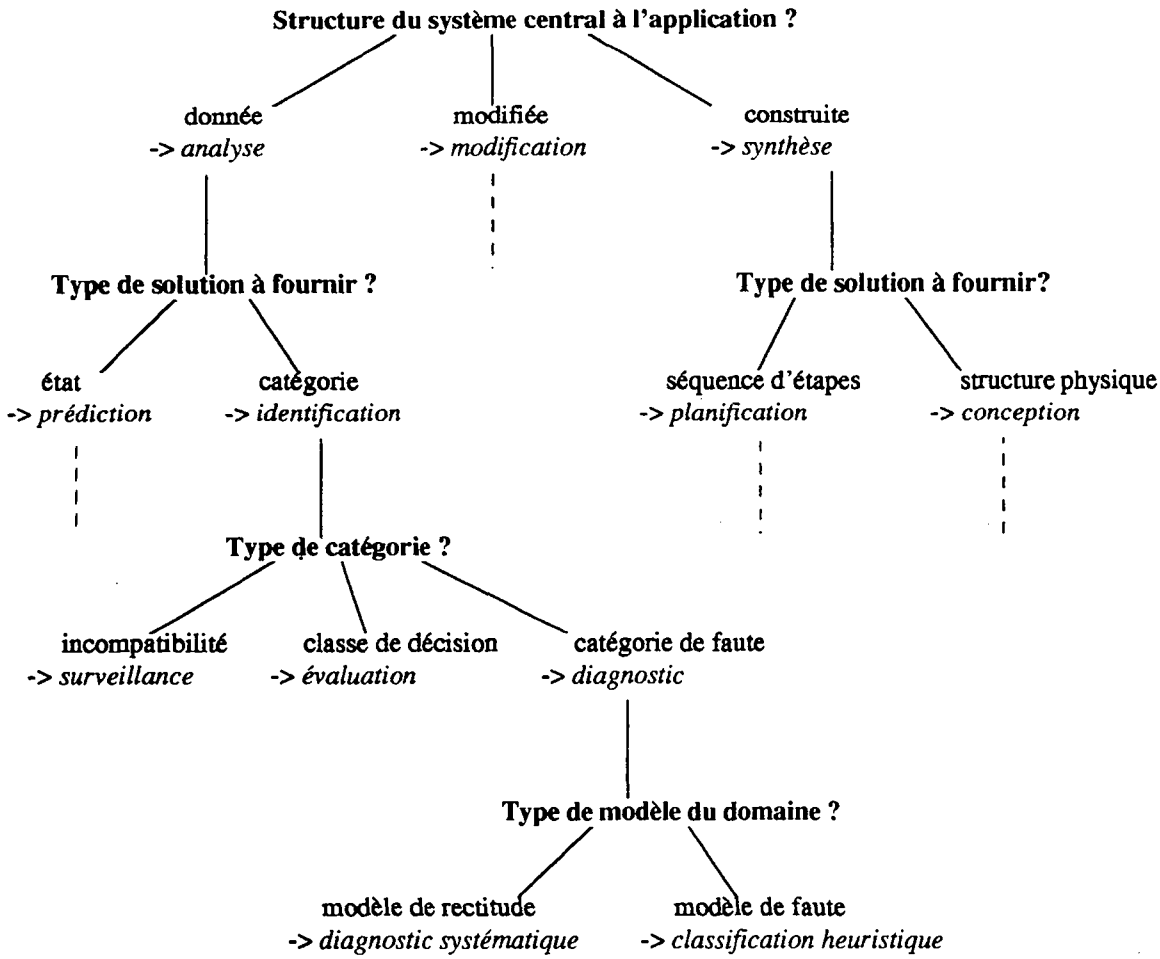
```
analyse_de_système
      identification  //d'une propriété ou d'un état
            classification
                  classification_simple
                  diagnostic //p 78
                        diagnostic_faute_unique
                              diagnostic-par-classification-heuristique //p 82
                              diagnostic_systématique //p 78
                                    localisation structurelle //p 79
                                    localisation causale //p 80
                        diagnostic_fautes_multiples
                  évaluation (de correspondance, d'adéquation, ...) //p 83
            surveillance //p 85

      prédiction //identification d'un état passé ou futur du système p 89
            prédiction_de_comportement
            prédiction_de_valeurs

modification_de_système //p 91
      réparation
      remède
      contrôle
            maintenance

synthèse_de_système //p 92
      transformation
      conception
            conception_par_transformation
            conception_par_raffinement //p 94
                  conception_par_raffinement_à_flot_unique
                  conception_par_raffinement_à_flots_multiples
            configuration //p 97
      planification //p 97
      modélisation //p 97
```

Remarquons que la tâche générique de localisation structurelle est le modèle de notre exemple de diagnostic systématique dans un système audio. De fait, le cogniticien utilisant la méthode KADS essaie toujours de **reconnaître** dans le discours de l'expert des tâches génériques de la bibliothèque. Une tâche réelle d'un domaine particulier est en effet très souvent une composition (parfois dynamique) de tâches proches des tâches génériques (ex: une tâche de réparation couplée avec une tâche de diagnostic).

Pour construire son Modèle d'Expertise, le cogniticien va donc sélectionner puis spécialiser (**remplir, enrichir et adapter** au domaine) et **combiner** des tâches génériques. La page suivante montre un arbre de décision partiel pour le choix des tâches génériques [Wielinga 92]. L'enrichissement peut consister à ajouter d'autres inférences ou remplacer celles existantes par des plus complexes ou encore faire intervenir (par des méta-classes supplémentaires) des connaissances spécifiques du domaine. Parfois cependant, aucune tâche générique ne convient et le cogniticien doit alors créer de toute pièces son Modèle d'Expertise (ou une bonne partie).

Dans [Wielinga 92], les auteurs notent que les modèles d'interprétation sont très utiles pour **planifier** et mener des entretiens, **interpréter** les données verbales recueillies, constituer une **documentation** de la base de connaissances ou encore pour générer des **explications**. Il semble que le cogniticien, même novice, ait peu de problèmes pour comprendre et utiliser ces modèles. Il en est de même pour l'expert qui peut ainsi mieux comprendre ce qu'on attend de lui (cependant, pour éviter les biais, dans la méthode d'acquisition proposée au chapitre 6, l'expert n'a pas connaissance des modèles d'interprétation essayés ou utilisés par le cogniticien).

Il est conseillé de lire l'annexe présentant les tâches génériques au cours ou à la fin ce cette section. Le chapitre suivant suppose que cette annexe a été lue.



Les tâches génériques servent donc de modèle initial pour guider le cogniticien dans l'analyse des données (d'où leur nom de «modèles d'interprétation» qu'ils partagent avec les modèles réels). Le cogniticien dispose en effet actuellement de très nombreuses techniques de recueil, d'analyse et de formalisation de la connaissance. Les modèles d'interprétation lui permettent de décomposer l'expertise par rapport à diverses classes de problèmes déjà connues et d'être ainsi guidé et de faire abstraction des détails.

Alliés à la diversité des modèles-étapes de KAD, les modèles d'interprétation permettent au cogniticien d'aborder la connaissance de l'expert, point de vue par point de vue. Et c'est un cadre plus ou moins universel de description de la connaissance qui est ainsi offert (en supposant qu'il n'existe qu'un nombre fini et petit de tâches génériques).

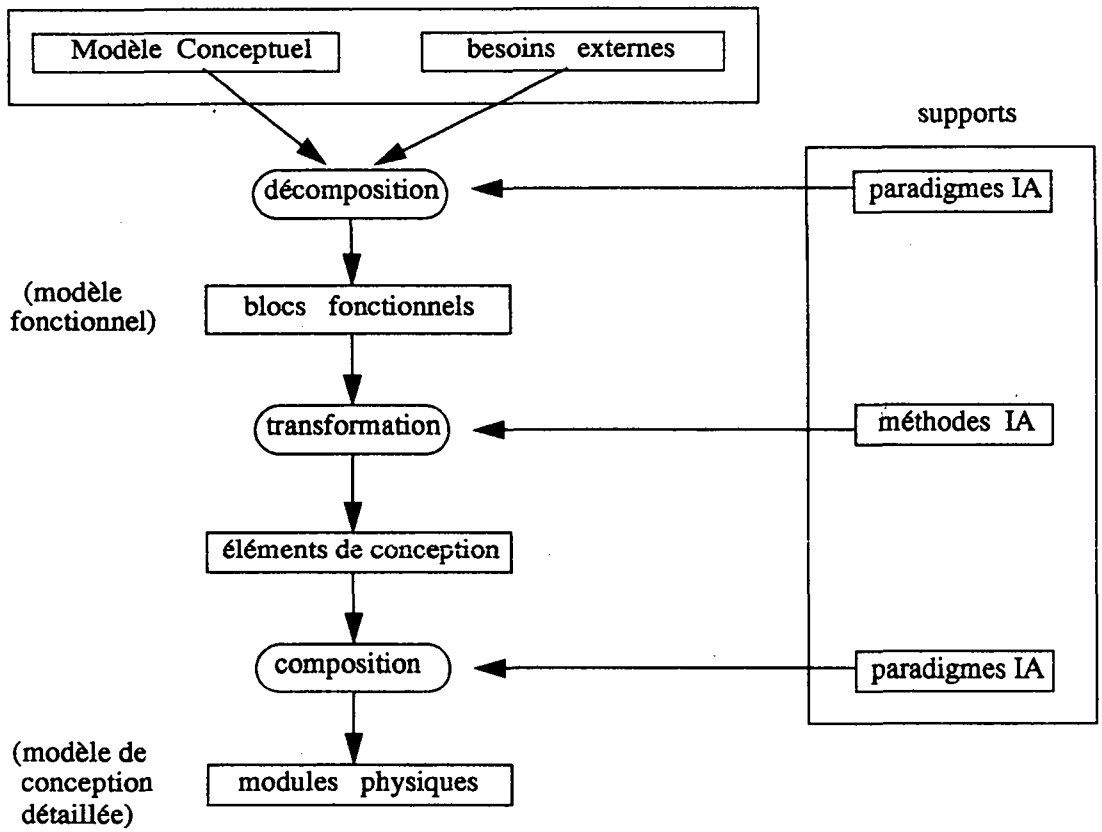
Un modèle d'interprétation suffisamment élaboré peut jouer un rôle similaire au prototypage rapide [Voss 90]. Bien qu'aucun engagement détaillé ne soit pris par rapport au langage d'implantation (extraction des connaissances plus fiable), certains outils (notamment dédiés au diagnostic) peuvent donner une idée du fonctionnement de modèles (tâche générique, Modèle Conceptuel).

Nous verrons en conclusion de ce chapitre l'évolution actuelle de ces modèles destinée à résoudre leurs ambiguités d'utilisation lors de l'extraction des connaissances.

3 La phase de conception

3.1 Processus

Trois étapes permettent de passer du Modèle Conceptuel au code tout en prenant en compte les contraintes externes (vitesse, logiciel, matériel) et utilisateurs (extrait de [Schreiber 88]).



- Une *description fonctionnelle* : cette spécification des fonctionnalités internes et externes du futur SBC (ce que requiert l'artefact final) s'obtient en transformant le Modèle Conceptuel en un ensemble de blocs fonctionnels représentant chacun une fonctionnalité distincte. Chaque bloc est caractérisé par son type de rôle (ex: résolution de problème, stockage de données, etc.) et ses relations avec les autres blocs : relations «consiste à» (spécifiant sa décomposition en d'autres blocs fonctionnels), relations «entrées/sorties» (précisant à quels autres blocs il sert ou de quels blocs il se sert), relations «contrôle» (définissant quels blocs il contrôle).

- Une *description de comportement* : sélection de méthodes pour réaliser les fonctionnalités de l'artefact et «d'éléments de conception» pour exécuter ces méthodes. Ce sont souvent des méthodes de l'IA comme les algorithmes de recherche, les classifications utilisant des relations «est-un», etc. Ce qui constitue la majeure différence avec la phase de conception des développements classiques. Voici quelques exemples de méthodes avec leurs éléments de conception associés.

méthode	type de rôle	éléments de conception
raffinement hiérarchique	résolution de problème	classifieur, subsumptions relations, définitions de classes, paires attributs valeurs
planification STRIPS	résolution de problème	propositions, sélecteur d'opérateur, opérateurs (ajouter-liste, détruire-liste, précondition)
parseur ATN	E/S de données	parseur, grammaires ATN, lexique, chaîne
ATMS	stockage de données	schéduleur, consommateurs, noeuds (donnée, étiquettes, justifications)

- Une *description de la structure de l'artefact* (comment les fonctionnalitées vont être implantées) : les éléments de conception y sont agrégés en modules physiques, bases de la conception détaillée et de l'implémentation. L'*environnement* (prolog, générateur de SE, etc.) fournit squelette d'architecture, méthodes et éléments de conception.

Pour guider la conversion du Modèle Conceptuel en modèle de conception (détaillé), il est utile de suivre un paradigme i.e un (ensemble de) principes. Voyons quelques exemples.

- «Minimiser l'interdépendance mais maximiser la cohésion» (des modules) est un paradigme de la programmation (ou conception) structurée. Les paradigmes conventionnels ne peuvent suffir au développement d'un SBC, il faut des paradigmes IA permettant de prendre en compte tous les aspects de la connaissance à implanter.
- Un paradigme très utilisé (EMYCIN, etc.) est celui de «système de production» préconisant pour modules physiques : une base de connaissance (contenant entre autres des règles utilisées en chaînage arrière), un moteur d'inférence (contenant les procédures physiques réalisant le chaînage arrière), une base de données et un module d'explication.
- Les systèmes construits avec le paradigme «recherche dans l'espace d'état» en tête (tel SOAR [Schreiber 88]) représente explicitement les différents états du processus de résolution (depuis l'état initial jusqu'à celui du but atteint). Chaque état peut être analysé et classé. Quelques méthodes de recherche dans cet espace d'état : profondeur d'abord, A-strar, MiniMax, etc.

Les recherches sur les paradigmes et leurs méthodes peuvent donc largement profiter aux développements des SBC.

Après cette présentation des «techniques» de la phase de conception, voyons plus précisément leur utilité.

3.2 Préservation de la structure du Modèle Conceptuel

Bien que de nombreux Modèles Conceptuels de KADS aient été implémentés avec des méthodes classiques (souvent parce que les générateurs de SE (GSE) utilisés fournissaient un ensemble limité de techniques d'IA), il est préférable d'utiliser une méthodologie préservant la structure du Modèle Conceptuel. En d'autres termes, le système final doit pouvoir relier les éléments du Modèle Conceptuel aux structures qui les implantent. Ceci pour plusieurs raisons ([Wielinga 92]).

- Les **explications** que peut fournir le système sont plus «profondes» car, au lieu de paraphraser le code, il peut générer des explications dans le langage du niveau conceptuel (tel [Clancey 84]). Un exemple, les auteurs de [Wielinga 92] ont développé un prototype de GSE opérationnalisant les systèmes de diagnostic. Son interface permet à l'utilisateur de tracer le raisonnement du SBC (généré), dans le vocabulaire du Modèle Conceptuel, à plusieurs niveaux, à travers : la visualisation de la structure de la tâche exécutée, la surbrillance de l'inférence dans la structure d'inférence quand l'inférence est exécutée, l'abstraction permise par les méta-classes et les termes de contrôles (i.e l'état courrant de la «mémoire de travail»), les connaissances du domaine utilisées par les inférences lorsqu'elles sont exécutées.
- La **maintenance** ou la mise au point est facilitée car il est possible de comparer précisément le fonctionnement réel d'un élément avec sa spécification et donc de détecter les erreurs ou omissions. Cela évite également la redondance des connaissances. Des buts similaires sont poursuivis par l'approche SEE (Système Expert Explicatif) qui garantit la préservation de la structure par une transformation automatique et la mémorisation de cette transformation.
- Les **outils** et techniques d'acquisition des connaissances (élicitation, génération de code; ex: grilles répertoires, algorithmes d'induction) peuvent être choisis en fonction du type de connaissances qu'ils génèrent (et non pour n'importe quel type). Ceci accroît l'interprétabilité et qualité des résultats. Cette approche est poursuivie dans le projet ACKnowledge [Heijst 92].

Un certain nombre de décisions doivent être prises pour implémenter le Modèle Conceptuel ([Schreiber 91]).

- Trouver une ou des techniques de représentation du domaine qui satisfassent les multiples points de vue et utilisations (résolutions, explications, etc.) sur celui-ci, tout en résolvant les éventuels problèmes de redondance (ou mise à jour). Ex: systèmes de production, recherches dans un espace d'états, parsing, matching ou classification.
- Synchroniser "l'algorithme" de la source de connaissance avec les "structures d'entrées/-sortie" des méta-classes. Des études ont été faites pour savoir quelles techniques choisir parmi tel ou tel grand groupe [Schreiber 89] ou à l'intérieur de tel grand groupe : classification hiérarchique [Goel 87], représentation logique et déduction [Reichgelt 86]. Comme les méta-classes sont naturellement typées, l'explosion combinatoire que l'on pouvait craindre - le Modèle Conceptuel ne spécifiant pas en détail le régime de contrôle des inférences - n'est pas un réel danger.
- Compte-tenu de l'ensemble des tâches (de transfert ou de résolution), il faut assurer :
  - le contrôle de leur exécution par agenda, tableau noir ou autre technique de planning;
  - la représentation et la mise à jour du contexte de l'exécution (systèmes de maintien de la vérité si non monotonie).
- S'il existe un niveau stratégie (généralement pour permettre de prendre en compte les préférences de l'utilisateur), utilisation de métarègles conditionnelles (influençant le contexte par l'activation de tâches) ou d'un tableau noir étendu ou encore d'un méta-système séparé du SBC.

L'utilisation d'un outil de développement peut grandement accélerer la phase de conception/implantation (cf par exemple [Karbach 88]). En contrepartie, beaucoup de choix seront faits automatiquement et le système final sera moins souple. Un système intermédiaire semble être une bonne solution [Voss 90].

4 *Apports et évolution de KADS*

KADS peut être utilisée seule ou en complément de méthodes classiques. Elle synthétise de nombreuses «bonnes idées» : séparation des phases d'analyse et de conception, prise en compte des méthodes de Génie Logiciel en les adaptant à la spécificité du développement des SBC, langage de modélisation de l'expertise, conservation du Modèle Conceptuel, bibliothèque de modèles d'interprétation.

Mais KADS tel qu'il a été présenté dans ce chapitre (i.e. KADS-I) est une première version qui a certaines faiblesses.

Tout d'abord, le langage de l'analyse est informel et les modèles d'interprétation proposés sont peu précis et, pour la conception au moins, insuffisants (cf annexe).
Le concepteur a donc beaucoup de libertés mais finalement peu d'aide : il doit lui-même compléter le langage et les modèles (utilisés).
Aussi, un certain nombre de recherches actuelles visent à offrir un langage de formalisation : (ML)2 ([Harmelen 92], FORKADS [Wetter 90], VITAL CML [Jonker 92], etc. Ce sont des langages très typés, proches de SML (et dans moindre mesure de ADA).
Ainsi, les modèles (de la bibliothèque et ceux construits par l'utilisateur) sont non ambigus et très précis tout en restant assez génériques. Ils sont instanciables sur différents domaines et peuvent être vérifiés (exécutés).
Mais ce sont des langages (très) informatiques, donc clairs seulement pour des informaticiens (il y a toujours des inférences, mais celles-ci sont décrites avec un langage formel).
[Schreiber 92] [Wetter 91] [Jonker 92] formalisent des modèles d'interprétation de diagnostic, montrant ainsi comment formaliser n'importe quel autre Modèle Conceptuel.

Deuxièmement, de nombreux autres modèles que ceux de la bibliothèque pourraient être utilisés pour les classes de problèmes visés et inversement, ces classes sont mal couvertes.
En fait, les tâches génériques ne sont pas au bon niveau de granularité : elles sont destinées à résoudre de trop larges problèmes avec des inférences trop précises. Il semble maintenant qu'il n'existe pas d'ensemble fixe de «méthodes de résolution», sinon d'une abstraction trop élevée pour être utile, mais une variation infinie de méthodes similaires dans leurs décompositions et leurs briques de base ([Gobinet 92]). L'école «blocs de construction» ([Steels 90] [Puerta 91]) cherche donc à offrir ces briques de base et les outils (langage, support) permettant de les combiner de façon appropriée à la tâche visée.
Une des voies de recherches de KADS-II s'inscrit dans cette voie de la souplesse et de l'adaptabilité, en travaillant sur des modèles d'un grain plus fin que les tâches génériques : les blocs d'inférences (génériques). Il a en effet été remarqué que les cogniticiens (utilisant KADS) construisent leurs Modèles Conceptuels en combinant des blocs de ce type.
La future bibliothèque de ces blocs devra être structurée comme un ensemble de modèles de construction (à appliquer à un modèle existant pour le rendre plus complexe) et non plus comme l'actuelle bibliothèque (cf [KADS-II 92Lib]).

Enfin, si le schéma général de l'utilisation des modèles d'interprétation pour aider à l'acquisition des connaissances, est donné dans KADS (identification du problème, ..., choix d'un modèle, ...), il reste que pour choisir un modèle, il faut des connaissances du domaine, et pour acquérir ces connaissances, il est utile d'avoir un modèle.
Les travaux du projet ACKnowledge (cf par exemple [Heijst 92]) apportent une solution : utiliser des «modèles directifs généralisés» extensions de modèles d'interprétation de KADS-I, possédant

des «sources de connaissances abstraites» qui sont petit à petit raffinées au cours du processus d'acquisition à l'aide de règles de réécriture et de connaissances déjà acquises.

Ainsi, si KADS-I a défini de très bons objectifs, il n'a pas vraiment donné les moyens de les atteindre. Cependant les diverses recherches en cours semblent être sur de bonnes voies pour cela. L'apport majeur de KADS est donc cette liste d'objectifs qu'il donne aux concepteurs qui suivent cette méthodologie. La très grande partie des descriptions de KADS confond d'ailleurs objectifs et méthodes à utiliser, ce qui rend ces descriptions assez floues et ambiguës. Pour minimiser ces défauts, un des buts dans l'écriture de ce chapitre et de l'annexe, a été de rassembler le maximum d'éléments concrets.

Chapitre 4 Explications : rôles et stratégies

Le chapitre 1 nous a montré quelles devaient être les qualités requises des connaissances du SBC pour de bonnes explications (structuration, abstraction, ...) et quelles connaissances devaient leur être associées (principes sous-jacent, points de vue multiples, ...). Pour cela, une typologie des opérations sur ces connaissances en vue l'explication a été donnée :
- justifier le lien entre données et résultats (type 1);
- donner des informations descriptives ou justifier les connaissances elles-mêmes (type 2);
- justifier la méthode de résolution mise en oeuvre, simuler ou critiquer un raisonnement alternatif.
Toute information (connaissance, trace, ...) devant pouvoir être filtrée ou synthétisée en fonction des désirs de l'utilisateur (notion de modèle utilisateur (MU)).

Les chapitres 1, 2 et 3 ont montré que ces spécifications rejoignaient celles des SE2 et des méthodologies actuelles d'acquisition et de modélisation de la connaissance.

Nous allons maintenant plutôt nous attacher à l'aspect communication de l'explication (identification des éléments pertinents, mise en forme, adaptation au contexte, choix entre différentes explications potentielles, gestion du dialogue, ...).

Ce chapitre, synthèse d'un domaine très riche en idées et directions de recherche a deux buts complémentaires :
- fournir une sorte d'expertise de l'explication (dans l'état actuel des recherches) qui sous-tende les modèles présentés au chapitre suivant;
- aider le concepteur à avoir une image assez nette des explications que son système devra offrir et des stratégies ou connaissances relatives à l'explication, qu'il devra acquérir puis implanter.

Nous verrons plus loin différentes stratégies explicatives. **Mais pour choisir entre celles-ci, un concepteur doit avoir un but clair du rôle et du contenu des explications qu'il veut que son système fournisse.**

1 Rôles

Le module d'explication peut par exemple servir :
- comme outil d'exploration de connaissances (utilisées par le résolveur ou de base de données);
- comme outil de mise au point (pour le concepteur);
- comme outil de validation (pour le concepteur ou des experts);
- comme outil de formation (pour des experts ou des futurs experts, des débutants);
- comme outil de décision (pour des spécialistes ou des débutants).

Il doit donc souvent (obligatoirement dans le dernier cas) présenter les connaissances utilisables par le résolveur de problèmes et rendre compte de ce qui a été fait ou va être fait par ce dernier pour augmenter la confiance de l'usager dans la capacité du résolveur à résoudre son problème.
Il est ainsi souvent amené à exposer une solution, synthétiser un raisonnement, justifier et commenter des choix (en précisant pour chacun son degré de certitude, son objectif, la stratégie ou tactique employée, le plan prévu pour l'exécuter), critiquer une autre solution. Dans un cadre d'enseignement, il peut aussi évaluer ou commenter des démarches, donner des indices pour démarrer, etc.

Dans le cadre d'une mise au point d'un SBC, les explications doivent refléter fidèlement les différentes étapes du raisonnement, en suivant l'implémentation réelle, pour mettre en évidence les connaissances correctement utilisées, celles rejetées à tort ou inversement. Dans les autres cas, la ligne d'explication doit plus ou moins s'éloigner de celle du raisonnement :
- simple synthèse, justifications, mise en valeur des points importants;
- indépendance totale vis à vis de la «trace» qui n'est alors pour le raisonnement explicatif qu'une source de connaissances parmi d'autres : messages à faire passer, buts et connaissances de l'utilisateurs, règles de communication, etc.

Bien que la génération d'explication soit généralement découplée de la résolution de problèmes, les connaissances nécessaires à cette dernière font toujours aussi partie des «Connaissances Utiles pour les explications» (seul [Wick 92] sépare domaine du module des explications et domaine du module de résolution). Ceci sera détaillé aux chapitre 5 et 6.

Hormis des contextes d'explications spécifiques, avec des utilisateurs très «typés», l'attente de l'usager est difficile à définir à l'avance, et le module d'explication devra donc s'adapter aux critères de satisfaction de l'utilisateur pour générer des explications utiles ou assimilables (ex: mettre l'accent sur la présentation des stratégies, les principes, les définitions formelles, etc.). Ces critères sont plus ou moins bien exprimés dans les requêtes. Nous verrons plus loin comment les rechercher, mais un des rôles des explications est de demander des éclaircissements sur ceux-ci.

Parallèlement à la satisfaction de ces critères, la production d'explications dépend également du rôle joué par le système dans la tâche de l'utilisateur. Trois situations peuvent être envisagées [Brézillon 92].

- Le système résoud automatiquement un problème pour fournir une réponse (rapide) et l'utilisateur ne fait que superviser l'exécution de la tâche. Le traitement d'alarmes entre dans cette catégorie. Les explications permettent dans ce cas de **comprendre** *a posteriori* le raisonnement tenu par le système au cours de la résolution, les choix qu'il a été amené à faire et les objets sur lequels ont porté le raisonnement.
- L'utilisateur est responsable de l'exécution de la tâche, et le système lui sert d'aide mémoire en lui fournissant les **informations** dont il a besoin ou la documentation qu'il souhaite compulser. Des SBC de ce type se trouvent par exemple dans le domaine de la législation. Les explications permettent à l'utilisateur de valider ses choix et de valider sa décision finale. Toutefois, ces explications n'influent pas directement sur l'exécution de la tâche.
- L'utilisateur et le système participent à la résolution du problème et la solution sera le résultat d'une **négociation**. La conception est un domaine où l'on trouve une telle situation. Les explications y sont fournies en cours de résolution de problème et peuvent éventuellement intervenir dans celle-ci. Elles constituent un moyen de communication entre l'utilisateur et le système.

Ceci peut donner lieu à trois sortes d'explications [Safar 92] (il n'y a pas correspondance directe, c'est pourquoi ces deux classifications sont utiles) :

- **réactives** : il s'agit de réponses directes à une requête de l'usager (avec appel d'un processus de résolution si celui-ci est nécessaire pour fournir le matériel de l'explication); ce sont les premiers types d'explications étudiés («qu'est ce que ...», «pourquoi ...», «pourquoi pas ...», etc.); on peut également rajouter dans cette catégorie les réponses déclenchées par des évènements (précisés par l'utilisateur dans une requête initiale);
- **intéressées** i.e. destinées à obtenir une contribution de l'usager; plus ce dernier est jugé compétent, plus la liberté de choix qui lui sera offerte sera grande et les mécanismes de décisions élaborés (et donc plus ces choix et leurs conséquences seront expliqués en détail);

on trouve de telles explications en apprentissage ou dans les commandes de modifications de système; ces questions doivent être posées au bon moment lors de la résolution de problème, sous peine de recevoir une réponse erronée ou un refus (du type «je ne sais pas»);

- **spontanées** soit pour occuper utilement l'usager en cas d'attente lors de l'exécution d'une commande et éviter qu'il puisse être surpris du résultat, soit pour éviter qu'il déduise de l'explication précédente de fausses informations (ou prévenir des questions ultérieures qui ont de grandes chances d'êtres posées), soit encore, pour le faire progresser dans un domaine.

Les processus explicatifs classiquement utilisés (réactifs) peuvent bien sûr aussi servir dans les types d'explications plus interactives mais ils doivent être adaptés, et contrôlés de façon adéquate. L'une des fonctionnalités à offrir dans ce cadre sont les interruptions. Une autre est de permettre «l'émergence de l'explication» à travers le dialogue. Ces deux points seront bien détaillés dans la section suivante. Il font un grand usage du modèle du modèle de l'utilisateur.

Un certain nombre de rôles de l'explication peuvent être remplis grâce à ce dernier. Le chapitre 1 en a déjà donné de nombreux exemples. Via le MU, le système connaît de façon assez générale les buts et le niveau d'expertise de l'usager : connaissances du domaine (et éventuellement connaissances sur le modèle du domaine et de résolution de problème utilisé par la machine).

Ce modèle peut être construit grâce à une librairie de modèles ou développé spécifiquement pour l'usager. Il peut aussi être construit ou amélioré sur la base des réponses et interventions de l'usager. Le système peut ainsi «espionner» l'utilisateur, enregistrer ses centres d'intérêt et la manière dont il recherche une information.
Ceci lui permet de déterminer les demandes les plus vraisemblables qui vont lui être faites, suggérer des solutions concurrentes qui peuvent ne pas avoir été envisagées, donner des exemples et proposer des raccourcis pour faciliter la recherche. L'usager doit cependant toujours pouvoir explorer librement la base de connaissances (et l'état des connaissances en cours de résolution et la base de faits).

Cependant, aussi complet soit il, le MU est insuffisant pour prévoir les réactions d'intérêt ou de rejet par rapport au discours explicatif ou au système expliqué. Seul le dialogue peut permettre d'ajuster l'explication, de la construire progressivement.

2 Spécifications du générateur d'explications

Maintenant que sont définis les divers rôles qu'un générateur d'explications peut remplir, voyons quelles sont les contraintes qu'ils imposent sur ce que doit contenir ce générateur.
Ainsi, le concepteur pourra-t-il, en fonction de ses objectifs d'explications, et en connaissance de cause, développer et implanter les différentes stratégies d'explications présentées dans la section suivante (et acquérir les connaissances nécessaires à ces stratégies comme expliqué dans les deux chapitres prochains).

Selon [Paris 92], pour être efficace et appropriée une explication doit être :
- informative i.e. contenir des informations que l'utilisateur ne connaît pas encore;
- cohérente i.e. les informations doivent être organisées et structurées;
- compréhensible i.e. être formulée dans un langage accessible;
- pertinentes i.e. permettre à l'interlocuteur de progresser vers la réalisation de ses buts.

Ces caractéristiques, largement étudiée en Langage Naturel (et peu dans les SE) imposent pour la génération d'explications de :
- disposer d'une grande variété de stratégie d'explication et s'adapter à l'utilisateur;
- utiliser les connaissances d'explication (linguistiques, pédagogiques, etc.);
- gérer le dialogue (retours de l'utilisateur, construction progressive de l'explication, etc.)

Disposer d'une grande variété de stratégie d'explication et s'adapter à l'utilisateur

De nombreuses stratégies d'explication sont possibles. Par exemple, pour expliquer un raisonnement :
- présenter la stratégie de résolution employée [Chandrasekaran 89] [Schulman 88] [Hasling 83];
- synthétiser le raisonnement suivi (en donner une explication globale) [Kassel 86];
- critiquer un raisonnement alternatif proche (explication négative) [Safar 89];
- produire un raisonnement analogue [Mittal 90];
- expliquer par comparaison [Alvarez 91];
- montrer des graphiques [Lambert 87].

Un système ne peut se limiter à une seule stratégie ni donc à un seul type d'explication et ce pour deux raisons [Lemaire 91].

- Une stratégie donnée ne s'applique que pour un contexte particulier, lequel comprend :
  - le type d'information à expliquer (pour cela, certains chercheurs associent une stratégie d'explication à chaque type de questions, certains dynamiquement; nous avons vu avec la typologie de Chandrasekaran, les principales classes d'informations à procurer);
  - les utilisateurs i.e. leurs connaissances (fournir par exemple des explications causales à un débutant et des explications structurelles à un expert), leurs préférences (définitions formelles, exemples, terminologie, explications graphiques, etc.), leur rôle social (conduisant à un discours poli, amical, etc.) et surtout leurs buts (pour interpréter les questions, sélectionner les réponses, etc.);
  - les explications précédentes qui aident à connaître l'utilisateur et ses buts, à comprendre les références implicites de ses questions (et inversement, à savoir ce qui peut être implicite dans la réponse);
  - le médium de communication (texte, images, son, ...), etc

- Il est bon de disposer de différents types d'explication pour une même requête :
  - pour donner différents points de vue à l'utilisateur sur un même objet ou raisonnement, améliorer une réponse mal comprise, etc. ;
  - pour avoir un discours varié et donc moins monotone (dans le cas d'un dialogue).

Il est donc nécessaire de munir le système d'un grand nombre de stratégies de discours et de techniques rhétoriques (section 3).

Utiliser les connaissances d'explication

Pour produire un texte naturel, de plusieurs phrases, les stratégies de discours doivent s'appuyer sur de nombreuses connaissances. Or, il n'existe pas d'expert en explication. Une certaine expertise existe, dispersée dans des domaines tels que **la pédagogie, la didactique, la psychologie, la rhétorique, la linguistique, l'ergonomie, etc.** Beaucoup de connaissances d'explication relèvent du **sens commun** : «ne pas répéter plusieurs fois la même chose de la même façon», «ne pas noyer l'interlocuteur sous un discours fleuve», etc.
L'expert du domaine de l'application peut posséder certaines heuristiques ou procédures d'explications (nous nommerons ces dernières des «schémas d'explication»; nous verrons plus loin qu'ils peuvent être implantés informatiquement par des «**schémas explicatifs**»). Cependant, il lui est impossible d'expliquer comment il réagit vraiment dans un dialogue réel (comment atteindre ou modifier les buts de discours qu'il s'était fixé).

[Lemaire 91a] et [Safar 92] proposent que les stratégies explicatives soit essentiellement développées dans le cadre de l'application et de ses futurs utilisateurs et conseille pour cela d'utiliser une démarche par prototypage rapide.
D'un autre côté, la méthode KADS conseille que le modèle de coopération soit indépendant du médium réellement utilisé. On rappelle que ce modèle précise les tâches de transfert entre l'usager et le système (donc aussi les explications).
Ceci n'est pas incompatible : la modélisation des explications peut être indépendante du médium et tenir compte des futurs utilisateurs.
Ceci est possible, car quoique les stratégies explicatives (présentées section suivante) soient dites textuelles, elle peuvent en fait être utilisées pour tout support du discours dont la nature est séquentielle (ex: le son). De plus, elles peuvent également inclure des images (exigeant un support spatial) dans leurs explications, voire éventuellement les construire : (étant donné un lexique et une grammaire, un générateur construit des objets dans le langage ainsi spécifié, que ce langage soit spatial ou séquentiel).
Les stratégies de l'explication réellement dépendantes du médium (donc de l'interface réelle avec l'utilisateur) sont à acquérir lors de la conception du SBC (de son interface). Mais, beaucoup de connaissances d'explication peuvent être receuillies indépendamment de l'interface réelle : **lexique, grammaire, sémantique et pragmatique** (i.e. comment utiliser les éléments précédents en fonction du contexte pour atteindre les buts du discours). Les chapitres 5 et 6 développent cette modélisation et cette acquisition dans le cadre de KADS.

Les recherches actuelles n'ont pas encore déterminé (mais est-ce entièrement réalisable) l'ensemble de connaissances d'explication **minimum** pour :
- donner les raisons de croire au contenu;
- évoquer les concepts appropriés, les faire comprendre (et retenir dans un cadre d'enseignement);
- ordonner les composants de l'explication pour mieux exprimer les concepts et relations exprimés;
- éviter les fausses inférences de la part de l'utilisateur, faciliter celles qui sont utiles;
- intégrer des nouveautés avec des choses déjà connues;
- etc.

Gérer le dialogue

Si le système ne possède qu'une seule chance de fournir une explication convaincante à son interlocuteur, il doit en posséder une représentation très détaillée et complète. Bien que de nombreux travaux aient été faits sur les modèles d'utilisateur, produire la bonne explication du premier coup est naturellement impossible (les SBC ne sont pas des systèmes élémentaires). Il vaut donc mieux se focaliser sur les **retours** de l'utilisateur et considérer que l'explication est une somme d'**échanges** avec ce dernier. [Moore & Swartout 89] définissent cinq caractéristiques que doit vérifier une telle approche de l'explication (approche réactive) :

- prendre en compte les retours de l'utilisateur (incompréhensions, interruptions, etc.);
- modifier l'explication si l'utilisateur indique qu'il n'est pas satisfait;
- répondre à des questions sur les explications précédentes (*«follow-up questions»*; selon Moore, les utilisateurs de systèmes d'aide à la décision sont obligés de poser de nombreuses questions «follow-up» pour comprendre les explications données);
- offrir des explications complémentaires même si l'utilisateur ne pose pas une questions «follow-up» bien formulée;
- utiliser l'information contenue dans le modèle de l'utilisateur s'il en existe un, mais ne pas en être dépendant.

Ceci impose de fortes contraintes dans la génération de l'explication :
- respect des règles de convention de dialogue;
- structuration du contenu adapté et support des interruptions;
- planification entre l'explication locale/réactive (l'explication ne se produit pas en une seule fois mais émerge d'une suite d'échanges avec l'utilisateur par un processus de négociation qui s'appuie sur le contexte du dialogue);
- capacité à s'adapter à différents types de dialogue (e.g. pour en changer en cas d'insuccès) et donc adapter automatiquement ses stratégies d'explications (cet objectif est à long terme).

Gérer les **interruptions** consiste à laisser la possibilité à l'utilisateur de rejeter l'explication ou plus fréquemment de demander des compléments d'informations intermédiaires. Les points d'interruptions offerts peuvent être après chaque mot, expression, phrase, paragraphe, etc.
[Brézillon 92] propose à l'usager d'introduire ces points après chaque concept. Ce dernier peut alors examiner le concept, ses relations avec d'autres concepts, inhiber les interruptions, changer le niveau de détail de l'explication en cours.
Ce dernier point montre que la gestion des interruptions doit prendre en compte des techniques de **dialogue** : changement de but ou de niveau de détail au cours de l'explication. Ceci implique de pouvoir commencer l'explication en gardant un niveau d'abstraction assez élevé pour les buts du discours non encore expliqués (et bien sûr de pouvoir revenir sur une partie d'explication fournie pour satisfaire un but d'explication passé).

Dans le cadre de la linguistique «interactionniste», de nombreux auteurs ont constaté que les interactions verbales en général présupposent une activité collaborative. Des recherches sur les explications s'appuient donc sur la **collaboration**, où les deux parties travaillent ensemble en maintenant une compréhension mutuelle, par opposition à la **coopération** où les deux parties peuvent travailler chacun de leur côté pour atteindre le but commun (définitions de [Baker 92]).

Selon [Baker 92], les explications collaboratives sont produites par la transformation successive d'un ensemble de propositions mutuellement comprises et acceptées, ensemble qui est lui-même le produit de ce processus. Voici les tâches de celui-ci (elles peuvent être implicites dans la conversation) :

- Etablir la compréhension mutuelle par rapport à des contributions et leur pertinence à la tâche d'explication : présenter, vérifier la compréhension, corriger une présentation, etc.
- Contribuer à l'élément courant de l'explication en le transformant successivement selon le domaine (avec accord éventuellement implicite après chaque modification) : généraliser, élaborer (facteur supplémentaire à prendre en compte), restreindre, reformuler, justifier, différencier (d'un autre cas), etc.
- Etablir un accord sur l'élément courant de l'explication et repasser à la première tâche pour aborder un sujet en relation ou pas avec l'élément courant.

Aucune des stratégies de base présentées ci-dessous ne modélise cette collaboration. Il est nécessaire pour celle-ci, comme pour tout dialogue, d'avoir **des stratégies spécifiques de la gestion du dialogue, collaborant avec les stratégies de détermination de contenu de l'explication qui travaillent en local sur des buts de discours**. Un niveau encore supérieur est également utile pour changer de sujet ou de type de dialogue, soit à la demande de l'utilisateur ([Cawsey 91a]), soit pour suivre un plan de session ([Chevalier 92]).

Enfin, dans le cadre de tout dialogue, chaque locuteur est lié par le «contrat de pertinence» [Karsenty 92] : n'utiliser dans ses communications que des éléments que l'interlocuteur est sûr d'avoir en tête compte-tenu du contexte du dialogue. La contrainte est certes trop forte pour pouvoir présenter un quelconque concept nouveau, et ainsi faire évoluer la conversation, mais le locuteur *doit utiliser «au mieux»* ces éléments. Il faut donc construire un modèle de cet «**environnement cognitif**» de l'interlocuteur et le faire évoluer à chaque échange (question ou réponse). Ceci permet non seulement de fournir aussi bien des **explications spontanées** (anticipation de question, etc.) mais aussi de minimiser le nombre d'éléments non «pertinents».

3 Stratégies d'explications

Nous avons vu ce que devait faire le générateur d'explication. Observons maintenant comment les principaux types de mécanismes actuels de génération de textes (ou discours) essaient d'atteindre ces objectifs. Ce sont des mécanismes **associatifs, d'exploitation, opportunistes, prescriptifs et restrictifs,** permettant de choisir et organiser le contenu d'une explication. A l'issue de cette description, la prochaine section montrera que l'implémentation de ces mécanismes (complémentaires) impose de nombreux **choix** et le chapitre prochain offrira un modèle pour aider à les **combiner** au mieux.

3.1 Approches schématiques

Ce sont les premières approches utilisées et se divisent en mécanismes «d'exploitation de structure» et ceux «d'imposition de structure».
Elles sont schématiques dans le sens où elles transfèrent un objet à l'utilisateur, sans réellement l'adapter au contexte, c'est à dire sans le transformer réellement (cas des techniques d'exploitation de structure) ou sans prendre en compte les retours de l'utilisateur sur l'explication (aucun des deux types ne gère le dialogue). Toutefois, les techniques d'imposition de structure peuvent produire des explications selon différentes stratégies.
Les approches schématiques sont maintenant utilisées (quoique transformées, dans le cas de l'imposition de structure) en **complément** d'autres techniques.

3.1.1 Exploitation de structures : les mécanismes de «traversées» de graphes

Les structures ici référées sont soit celles de la base de connaissance, soit celles générées comme traces (mémorisation des choix et des actions) au cours du raisonnement. Ce peut donc être des hiérarchies de concepts, des réseaux causaux, etc. Le générateur d'explication suit sélectivement les liens du graphe (trace ou bien structure du domaine) et produit ainsi à la fois la structure et le contenu de l'explication.

La plupart des modules d'explications des SE1 n'appliquent que ce principe. Nous avons vu au chapitre 1 que pour être relativement compréhensibles, de telles explications qui suivent la ligne du raisonnement devaient s'appuyer sur une trace suffisamment abstraite. Rappel : ceci peut s'obtenir par un code bien structuré (décomposition des tâches en sous-tâches, chaque tâche poursuivant un but relativement abstrait et ayant un contrôle explicite). Un moyen proposé par [Swartout 83] pour avoir des explications suffisamment abtraites sans perdre le bénéfice de la rapidité d'un code adapté au domaine est de générer le code en instanciant sur le domaine des principes de plus haut niveau et de mémoriser cette génération. Le module d'explication peut alors retrouver d'après la trace, les principes appliqués. Dans la méthodologie KADS, la conservation de la structure du Modèle Conceptuel lors de la conception permet d'appliquer la même démarche.

Dans la mesure où il est plus facile pour l'utilisateur lisant ou écoutant l'explication, de reconstruire un modèle cohérent (du message ou contenu de cette explication) si chaque élément est introduit en relation avec ceux qui le précèdent (exploitation de la structure relationnelle de la connaissance), alors ces mécanismes sont bien adaptés. Exemples : parties d'objet introduites *en relation avec* leur contenu ou leurs parties adjacentes; descriptions de processus suivant des liens causaux ou temporels.

Ces mécanismes servent essentiellement à organiser localement des éléments d'informations ou à

composer des choix locaux. Ils ne peuvent en effet être utilisés pour une organisation globale qui dépend d'un niveau d'abstraction plus élevé que les liens suivis (ex: «décider de présenter» et «présenter un modèle structurel cohérent d'un objet» avant de «décrire un processus qui affecte cet objet») [Suthers 91].

Les méthodes suivantes peuvent aider à organiser à un niveau plus élevé les informations produites par les techniques de «traversée» de graphe et à les adapter à l'utilisateur.

3.1.2 Imposition de structures

Il s'agit de trois techniques de planification de texte (discours) : le remplissage de schémas rhétoriques [McKeown 85], le raisonnement sur les croyances des interlocuteurs [Appelt 85] et la composition de relations rhétoriques [Hovy 88].
On rappelle, à titre culturel et dans une moindre mesure pour aider à comprendre les différents emplois ci-dessous de l'adjectif rhétorique, que la «rhétorique» comprend les figures de pensée (tours de pensée indépendants de l'expression tels l'antithèse, l'interrogation, l'énumération, la gradation, la réticence, la périphrase, l'hyperbole, etc.) et les figures de mots (qui détournent le sens des mots comme l'ellipse, l'inversion, la métaphore, l'euphémisme, etc.).

Rappelons que le résultat des mécanismes de planification de texte n'est pas du texte, mais une spécification formelle de celui-ci, par exemple, sous la forme de graphes conceptuels [Thomas 92]. Cette spécification doit ensuite être fournie à un générateur de phrases (voire de graphiques ou de sons s'ils sont spécifiés) qui prend en compte les possibilités d'une grammaire et d'un lexique pour satisfaire au mieux les spécifications (tout langage a une grammaire et un lexique, les graphiques peuvent donc en avoir).
Cette phase de génération de «phrases», triviale ou complexe suivant les ambitions du concepteur, permet de ne pas compliquer le processus de planification en prenant en compte un si bas niveau d'abstraction. [Bateman 89] conseille une étape intermédiaire supplémentaire pour émettre des contraintes sur la grammaire et le lexique à utiliser en fonction du type de l'interlocuteur (tournure de phrases, expressions, vocabulaire, etc.).

Souvent identifiés à partir de l'analyse de textes ou de discours, les **«schémas»** sont des structures fixes composées de **prédicats rhétoriques** génériques (ex: identification, analogie), traités dans un ordre pré-déterminé. Chaque objectif de discours est représenté par un ou plusieurs schémas, et les connaissances nécessaires sont obtenues par des fonctions associées à chaque prédicat. Ils sont **associés à des types de questions prédéfinis et instanciés de manière contextuelle**. Ci-dessous, pour exemple, le «Constituency schema» de McKeown, qui sera utilisée pour toute définition de concept - McKeown ayant remarqué que les explications humaines décrivent souvent les objets de la façon suivante : - identification de l'objet en tant que membre d'une classe générique;
- présentation de ses attributs et des ses constituants;
- description de chaque constituant;
- ajout d'informations attributives ou analogiques.

```
Identification/Attributive
Constituency
Cause-effect* / Attributive* /                    // le «/» indique une alternative
   {Depth-identification / Depth-attributive        //  et «*» une répétition
      {Particular-illustration / Evidence}         // et «+» une répétition non vide
       {Comparison ; Analogy / Renaming} }+
{Amplification / Explanation / Attributive / Analogy}
```

Les faiblesses d'un schéma proviennent de son aspect figé ou «compilé» :
- il ne rend pas compte de structures de discours propres à de nombreux domaines;
- les sous-buts poursuivis ne sont pas explicites (ni leurs liens pour former un texte cohérent atteignant l'objectif global) et il est donc impossible de raisonner dessus; si l'explication est incomprise, il est impossible de savoir quelles parties n'ont pas atteint leur but et la seule solution est d'essayer un autre schéma (d'où l' inaptitude de cette technique au dialogue).

On peut ainsi remarquer que les schémas rhétoriques sont des versions étendues et explicites, des stratégies d'explication utilisées dès les premiers modules d'explication de SE. Voici ceux de Néomycin (mais il s'agit là plus d'un exemple de «traversée de graphes» sur une trace relativement structurée, que d'une imposition de structures) :

```
Pour une question «why ?» posée lors d'une requête d'une valeur :
  afficher le but poursuivi par (la tâche qu'exécutait) la métarègle ayant déclenché la requête
  afficher les prémisses de cette métarègle (instanciées si l'utilisateur le préfère)

Pour une question «why ?» posée sur la réponse à la question «why ?» ci-dessus :
  afficher le but poursuivi par (la tâche qu'exécutait) la (méta-)métarègle ayant invoqué
                    (la sous-tâche qu'exécutait) la métarègle ayant déclenché la requête
  afficher les prémisses de cette (méta-)métarègle

Pour une question «how ?» posée à propos d'une tâche éventuellement en cours et que le module
      d'explication a déjà mentionné (au cours d'une réponse à une question «why ?» ou «how ?»)
  citer les métarègles utilisées pour exécuter la tâche en signalant si elles sont achevées,
                                                    en cours, ou encore à appliquer;
  (ainsi, la réponse est d'autant plus abstraite que la tâche mentionnée est de haut niveau)
```

Le raisonnement sur les **croyances des interlocuteurs** [Appelt 85] comporte ce qui manque aux schémas : dans cette approche, chaque opérateur de planification indique ses effets sur les croyances de l'utilisateur. Par contre le raisonnement est tenu au niveau des actes de language car aucune connaissance rhétorique n'indique comment composer ces actes pour former un texte cohérent pour atteindre un objectif global. Par ailleurs cette approche est très coûteuse en temps machine et a donc été utilisée principalement pour générer des textes d'une ou deux phrases contenant des expressions anaphoriques.

Plus récemment, [Hovy 88] a utilisé les **relations rhétoriques**, sous la forme d'opérateurs de planification associés à des conditions d'application, pour organiser en un texte cohérent des connaissances sélectionnées à priori. Ces relations sont basées sur la *Rhetorical Structure Theory* (RST) (Mann et Thompson 1987, 1988) spécifiant qu'un texte est cohérent s'il existe une relation RST (ex: motivation, cause, etc.) entre toutes différentes parties du texte. Chaque relation est définie par un noyau (représentant la partie du texte contribuant le plus au but de l'interlocuteur) et un ou plusieurs satellites (aidant le noyau à réaliser ce but). De plus, des conditions doivent être vérifiées pour que la relation soit applicable. Hovy a rajouté, pour pouvoir construire des textes de plusieurs phrases, des points de développement indiquant les autres relations éventuellement incorporables.

Cette approche présente deux inconvénients :
- il n'y a pas bijection entre les relations rhétoriques et les buts de discours; ainsi, bien que le système puisse reconnaître la relation entre deux phrases (ou deux parties de texte), il ne sait pas le but communicatif qu'elles cherchent à atteindre (d'où inaptitude au dialogue);
- les faits à présenter doivent avoir été choisis auparavant (par une autre technique).

De manière générale, les mécanismes d'imposition de structure laissent le choix des connaissances pertinentes pour l'explication, à d'autres mécanismes - une exception : les arbres de topiques de [Carcagno 89] utilisés pour déterminer un contenu qui est ensuite structuré.
Cela les rend inadéquats pour la génération de structures pédagogiques pour certaines explications [Suthers 91]. Exemple : fournir avant une information de réponse à une question, d'autres informations permettant de la comprendre. Ces informations de «contexte» ne doivent être incluses que si elles sont supposées ignorées de l'interlocuteur (compte-tenu de l'historique du dialogue et du modèle de l'utilisateur). Des structures fixes ne peuvent modéliser les relations dynamiques déterminant le rôle et la place de ces informations (de contexte). Plus généralement, il semble que le choix des éléments à intégrer dans un texte et leur organisation, ne peuvent être découplés car ils exercent l'un sur l'autre des contraintes ([Lemaire 91]).

La présentation de ces trois approches élémentaires est néanmoins utile dans le cadre d'une aide à un concepteur de module d'explication (objectif du chapitre) pour au moins trois raisons :
- montrer les inconvénients ou les manques de telles méthodes;
- ces approches sont simples (surtout les schémas) et le concepteur peut décider que, compte-tenu de ses objectifs d'explication, elles sont suffisantes (ex: réponses courtes et ayant peu d'alternatives significatives); il peut aussi décider de les appliquer pour structurer les parties d'un contenu pré-sélectionné et grossièrement structuré;
- certaines approches plus élaborées de la section suivante relèvent d'un mixage de ces trois mécanismes; il faut donc comprendre les raisons de celui-ci; par ailleurs, le «dosage» varie suivant les auteurs (bien que l'approche relations rhétoriques serve de base); le concepteur pourra choisir le sien en fonction de son application et des contraintes sur son module d'explication.

3.2 Approches planification de contenu

Les approches ci-dessous déterminent en même temps structure et contenu, mais certains préfèrent procéder en deux étapes (ex : [Zukerman 91]). Comme c'est la recherche du contenu qui fait défaut aux techniques de la section précédente, c'est cet aspect qui nous intéresse ici.
De plus, ces stratégies permettent de gérer l'explication en tant que dialogue (du moins en partie; les interruptions ne sont pas traitées).

La première démarche a consisté pour cette gestion, à construire l'explication en s'appuyant sur l'objet à expliquer et en transformant et adaptant cet objet au contexte. Elle est issue de de la communauté «systèmes experts» où l'objet est le raisonnement, ce qui justifie une telle démarche **ascendante** [Lemaire 92]. Les systèmes BLAH [Weiner 80], C.Q.F.E. [Kassel 86], Pourquoi-Pas? [Safar 89], PROSE [Dominguez 89] et KNIGHT [Lester 91] illustrent cette stratégie.

Par la suite, une autre approche a surtout travaillé sur l'explication de connaissances factuelles (i.e. la partie domaine dans KADS) concentrant ainsi son effort sur la tâche d'explication (et son aspect communication). Le raisonnement d'explication s'exprime alors en termes de buts ou d'actes de communication qu'il s'agit de décomposer. La construction de l'explication ne dépend plus d'une structure figée mais relève d'une planification **descendante** dynamique du contenu explicatif (*«plan-based explanation»*) [Cawsey 89] [Moore 91], [Maybury 91] (ce dernier a adapté l'approche à l'explication des raisonnements).

Ce plan est construit par des opérateurs de planification qui décomposent des concepts abstraits (initialement un but de discours) jusqu'à obtenir des primitives de discours (éléments textuels). L'**encadré 2** sur le générateur de texte EES [Moore 91] **détaille** ceci. Voici également, à titre d'exemple, l'opérateur *«How-it-works»* de Cawsey (destiné à faire comprendre comment fonctionne un composant).

```
(defplan how-it-works (device)
       :constraints    ((greaterp expertise-level novice-level))
       :preconditions   ((structure (device)))
       :subgoals        ((teach causal-behaviour (device))
                        ((teach what-it-does (device)))
       :template        (« how the » (getslot device 'name) « works »))
```

Lorsque plusieurs opérateurs sont candidats, des heuristiques permettent de faire le choix de celui à appliquer. Voici certaines des heuristiques de Moore :

- éviter les opérateurs qui font des hypothèses à propos des croyances de l'utilisateur
- préférer les opérateurs les plus spécifiques
- préférer les opérateurs qui utilisent un concept qui est déjà intervenu dans le dialogue.

Le plan de l'explication se construit donc progressivement en sélectionnant à chaque étape les meilleurs opérateurs («meilleurs» compte tenu du contexte). Outre la flexibilité qui en résulte, le générateur peut répondre à une question sur (une partie de) l'explication donnée en raisonnant sur les parties qui n'ont pas atteint leurs buts (les informations sur la construction étant conservées). C'est une voie vers la construction de l'explication via un dialogue.

Pour obtenir un module d'explication capable de couvrir un grand nombre de questions, de disposer de plusieurs types de réponses possibles, adaptées à l'utilisateur, et de supporter le dialogue, Moore & Paris ont utilisé un mécanisme de planification descendante dans lequel les plans représentent soit un but de discours, soit une relation rhétorique (relation RST). Chaque plan (un terme plus exact par rapport au texte de la section en cours serait opérateur de planification) comprend une ou plusieurs étapes qui sont elles-mêmes des buts ou des actes primitifs du discours.
Un texte est formé en spécifiant l'objectif du discours : le plan jugé le plus apte à atteindre cet objectif est exécuté, définissant à sont tour d'autres sous-buts, etc. Le processus se termine losrque tous les plans ont été déployés en actes primitifs (ceux-ci seront traduits en anglais). L'arbre de la génération indique les relations rhétoriques entre les buts (donc leur interaction). Il est conservé pour la génération de texte et pour répondre au questions sur le texte (le rassemblement des actes en une structure se rapproche d'un schéma, mais les relations entre ces derniers sont ici explicites, ce qui permet le raisonnement).
Chaque plan comprend :
- un effet : but de discours tel «arriver à l'état où l'utilisateur est convaincu d'une conclusion» ou relation rhétorique comme la motivation;
- une liste des conditions d'applicabilités portant sur les bases de connaissances ou le modèle de l'utlilisateur. Lorsque celui-ci est insuffisant (incomplet) le générateur fait des hypothèses sur la validité des conditions (elles sont stockées dans l'arbre de génération);
- une liste d'étapes permettant d'accomplir l'objectif du plan (un noyau et des satellites optionnels)

Voici un plan destiné à faire réaliser un acte par l'utilisateur (l'acte est une solution trouvée par le système pour satisfaire un des *buts de l'utilisateur* ; le *but du système* est de faire réaliser cet acte; si l'utilisateur le demande, la justification de cette solution est donnée (cf «persuader»))

```
EFFECT: (GOAL ?hearer (DO ?hearer ?act))
CONSTRAINTS: nil
NUCLEUS: (RECOMMEND ?speaker ?hearer ?act)          //recommender l'action
SATELLITES: ( ((COMPETENT ?hearer (DO ?hearer ?act)) *optional*)
              ((PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act))) *optional*))
//optionnellement, si l'utilisateur peut réaliser l'action, le persuader de la faire
```

Et voici l'un de ceux destinés à le persuader :

```
EFFECT: (PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act)))
CONSTRAINTS: (AND (GOAL ?speaker ?goal) (STEP ?act ?goal) (GOAL ?hearer ?goal))
       //si l'acte est une étape de but(s) commun(s) à l'utilisateur et au système
NUCLEUS: (FORALL ?goal (MOTIVATION ?act ?goal))  //motiver l'action par ce(s) but(s)
SATELLITES: nil        //«motivation» est une relation rhétorique
```

Le plan de l'explication est ainsi raffiné et inclut en final des connaissances du domaine et de la résolution. Si l'utilisateur demande que le texte soit plus clair (question «huh ?»), le générateur revient sur des suppositions qu'il avait faites concernant les connaissances de l'utilisateur (cf liste des conditions d'applicabilités) ou essaie d'autres stratégies.

Encadré 2. Le générateur de texte de EES ([Moore 91])

[Cawsey 89] gère le problème des **informations de «contexte»** (i.e. fournir si besoin, avant une information de réponse à une question, d'autres informations permettant de la comprendre) grâce aux préconditions de ses opérateurs de plans : les informations ne sont bien incluses que si l'utilisateur est supposé ne pas les connaître. De plus, pour les traitements ultérieurs, le générateur sait de quelles informations il s'agit, puisqu'il les sélectionne lui-même.

Cependant, les stratégies explicatives étant abstraites, elles ne spécifient pas exactement quel contenu sera sélectionné et ce dernier peut donc contenir des concepts non anticipés qui ont besoin d'être expliqués [Suthers 92]. Un mécanisme **opportuniste** du type «critiques dirigées par les données» est ainsi **utile dans tout modèle d'explication planifiant avec des stratégies abstraites.**

3.3 Nécessité de mécanismes opportunistes

La construction descendante suppose une connaissance non négligeable des résultats attendus de chaque opérateur et de leurs interactions, puisque leurs choix ne sont pas remis en cause. Une amélioration consiste donc à permettre ces modifications et à retarder les choix faits durant la construction
Les corrections de l'objet peuvent être faites après sa construction ou pendant celle-ci, i.e. par alternance des phases de construction et de correction. Cette alternance est tout indiquée dans une construction par raffinement. Ces deux principes, **retardement des choix et amélioration des essais**, sont souvent utilisés dans les processus de construction complexes (selon [Pitrat 90], les techniques d'amélioration sont plus faciles à trouver que les techniques de construction directe). Ces principes peuvent servir quelles que soient les techniques utilisées pour construire l'explication initiale.

Parmi les mécanismes opportunistes (i.e. dont les actions ne peuvent être «programmées» à l'avance) on peut distinguer :
- les stratégies dirigées par les données pour **modifier et augmenter** le plan de l'explication
- les stratégies dirigées par les données pour **déterminer la structure** du contenu

Les **premières** sont utiles pour :

- pallier aux insuffisances du plan de l'explication généré par des stratégies abstraites;
- examiner le contenu sélectionné en relation avec le modèle de l'utilisateur pour prévenir des inférences erronées ou l'incompréhension de ce dernier (justifier les concepts complexes, détailler ceux mal maîtrisés par l'utilisateur, illustrer ceux inconnus; s'il existe un concept proche de celui à expliquer et connu par l'utilisateur, tenter une explication par analogie; placer de façon plus appropriée certaines informations (ex: placer les exemples après les définitions, changer «A et B et ... implique D» en «D parce que A et B et ...»), etc.);
- filtrer et regrouper différentes informations (suppression des éléments redondants, utilisation de concepts regroupant des éléments similaires, etc.) générées par l'expansion de buts de discours indépendants; une explication longue doit être divisée en sous-parties reliées par des méta-commentaires (transitions ou commentaires sur le discours lui-même comme par exemple : «le point suivant est délicat») ou utiliser des marqueurs sémantiques.

[Lemaire 91] distingue des «connaissances de contrôle» nécessaires pour contraindre l'application de ces stratégies ou les départager en cas de conflits. Par exemple :
- limiter le nombre de suppositions faites à propos des connaissances de l'utilisateur ([Moore 91]);
- l'explication proposée à un débutant ne doit pas être trop longue.

[Mooney 91] a montré que les explications étendues (multi-paragraphes) ont dans certains domaines des plans d'organisation déterminés par le contenu (ex: temps, espace, évènements, facteurs) et des critères de segmentation correspondants (ex: «clusters» temporel, spatial, évènements causaux) au plus haut niveau de l'organisation. Il suggère donc d'utiliser un mécanisme opportuniste travaillant sur le noyau initial de spécifications de contenu avant d'appliquer des méthodes descendantes pour la sélection et la structuration du contenu à l'intérieur d'un segment. Ceci pour illustrer le **second** type de stratégies donné. Les mécanismes de «traversée» de graphes peuvent également rentrer dans cette catégorie lorsqu'ils servent à (re)structurer des éléments localement.

Les mécanismes de cette section collaborent avec le mécanisme central de recherche de contenu pour le guider ou le corriger. Ceux de la section suivante créent des contraintes sur la sélection de ce contenu.

3.4 Utilisation de la «perspective» pour la recherche de contenu

Déterminer la perspective pour l'explication, c'est choisir les propriétés et relations qui seront mis en avant et les concepts qui serviront de base pour comprendre le phénomène à expliquer.
Ce choix permet à l'explication d'être plus **compréhensible** (utilisation du modèle de l'utilisateur, de l'historique du dialogue, ...) et plus **adéquate** (prise en compte des buts de l'utilisateur, l'utilisation optimale de tous les éléments de réponse disponibles (domaine, interface, ...), etc.).

Les **mécanismes d'association** exploitent les relations entre les concepts pour déterminer cette perspective.

La familiarité d'un concept (recherche de compréhension) peut ainsi être évaluée par ses relations avec des concepts que l'utilisateur a utilisés. Par exemple, un usage approprié de «potentiel électrique» suggère que «champs de force» pourra vraisemblablement être compris, ce qui est moins sûr, si rien de plus spécifique que «électricité» n'a été mentionné. Ce style de recherche pourrait reposer sur des liens valués entre concepts (et plus généralement, sur certaines relations sémantiques) [Suthers 91].

Dans le même esprit, [Lester 91] crée un réseau intermédiaire comprenant : les concepts devant être expliqués, les concepts familiers à l'interlocuteur qui sont liés (de près ou de loin) avec les concepts devant être expliqués, et les relations entre tous ces concepts (les relations entre concepts familiers et concepts à expliquer contiennent des concepts intermédiaires).
Pour permettre la création de ce réseau intermédiaire, des «vues» et des «types de vues» fournissent des réseaux cohérents sur la base de connaissances. Lester distingue les types de vues «*hiérarchiques*» (comment un concept rentre dans une hiérarchie), *fonctionnelles* (le rôle d'un objet dans un processus), *structurelle* (parties d'un objet ainsi que les objets dont il fait partie) et *modulatoire* (comment un objet ou processus affecte ou est affecté par un autre objet ou processus).
Etant donné un concept à expliquer, différentes procédures de connexion utilisent ce types de vues pour déterminer les concepts pertinents familiers à l'interlocuteur (utilisation d'un modèle de l'utilisateur).

[McCoy 89] fournit un exemple d'association dans un but d'adéquation : corriger des conceptions erronées en se basant sur des attributs et des objets mentionnés dans le dialogue précédent. Elle utilise pour cela des collections d'attributs valués (représentant des perspectives spécifiques au domaine) comme filtres quand elle examine le modèle de l'utilisateur et choisit les réponses possibles.

Une fois choisie, la perspective influence la sélection du contenu en donnant des «**préférences**» lors des points de choix résolvant ainsi des conflits que des mécanismes prescriptifs (comme la planification descendante) arrivent mal à gérer seuls. Ces préférences peuvent évoluer dynamiquement, mais restent actives tout au long de la sélection [Suthers 91].

4 De multiples choix

Nous venons de voir les grandes lignes de la génération de l'explication. Aucune des techniques présentées n'est suffisante à elle seule. Il faut donc les combiner et assurer leurs divers contrôles et collaboration. Voyons les problèmes que cela pose. Le lecteur pourra alors mieux comprendre le modèle proposé au chapitre suivant, qui combine ces techniques.

Prédicats, relations, opérateurs, etc..

Tous les types de primitives sont importants et peu de recherches utilisent de façon pure ces éléments «rhétoriques» : si les prédicats sont bien indépendants du domaine, les opérateurs de planification de Cawsey (ex: «*How-it-works*») en sont plus proches. Pour certaines applications, les caractéristiques des objets du domaine sont importants pour diriger la structuration. Aussi, la création d'opérateurs ou de schémas dépendants du domaine simplifie le problème. Cependant, ces opérateurs devraient être séparés de ceux plus généraux ou identifiés comme leurs instances.

De façon similaire, les approches basées sur les relations rhétoriques devraient clarifier les relations entre les buts rhétoriques et les relations entre sections (de texte) ainsi que ce qui peut réellement servir en tant que but rhétorique [Cawsey 91].

Qu'inclure dans le contenu de l'explication avant de commencer à le structurer ?

Ou encore, à quel moment de détermination de contenu interviennent les méthodes de planification ? Quelle est l'importance du travail préalable à opérer sur les traces et les connaissances disponibles (sélection des concepts et stratégies pertinents en fonction du contexte) ?
Ceci bien sûr dans le cas où un texte est généré, car l'explication peut par exemple consister à construire un arbre de déductions ou de tâches et à lui permettre de naviguer dedans ou de demander un degré de détail supérieur.

Selon [Cawsey 91], il n'est pas nécessaire pour une explication basée sur le dialogue de savoir exactement ce qui va être dit avant de commencer à structurer puisque la perspective (MU) peut être déterminée grâce à l'expansion des buts (remarque: ceci ne veut pas dire que la phase d'adaptation des diverses connaissances à l'état courant du dialogue ne soit pas nécessaire).
Plus généralement, il serait inutile de trop spécifier le contenu lorsque les techniques de planification peuvent le faire. Utile pour ne pas rendre la planification trop complexe, la détermination de la perspective doit suivre des buts de haut niveau pour ne pas empiéter sur le travail du planificateur et pour s'adapter aux situations de dialogue (le contexte et les buts du discours pouvant alors évoluer au cours de l'explication).

Cependant, les connaissances ont parfois des plans d'organisation (conceptuels) suffisamment cohérents pour être exploités et ainsi fournir une perspective assez stricte aux techniques de planification. Il semble que la planification ne puisse gérer seule des explications étendues (nombreux paragraphes) et il est certain qu'elle ne peut gérer un vrai dialogue (plus que quelques interruptions ou questions «follow-up»).

Interruptions et dialogues

Dans ces deux cas, des (débuts d') explications doivent pouvoir être donnés alors que la suite n'est que vaguement spécifiée. Ceci est trivial avec les mécanismes descendants (en profondeur d'abord), c'est possible mais inutile avec les schémas puisque de toute façon, il ne peuvent raisonner sur leurs buts. Il faut toutefois pour la planification descendante, un mécanisme spécial pour revenir en arrière et redévelopper certaines parties.

Pour la gestion du dialogue même, [Cawsey 91a] propose dans un souci de simplicité d'utiliser le même mécanisme de planification descendante que pour la détermination de contenu. Seuls les opérateurs changent. Certains sont de haut niveau (gestion générale des échanges et des interruptions) tandis que d'autres sont plus proches de l'application (ex: enseignement, conseil, etc.).
Parmi les premiers, elle distingue quatre types : la *transaction* sur un sujet (unique), l'*échange*, le *mouvement* dans cet échange et l'*acte* («une transaction est normalement composée de séquences particulières d'échanges, d'échanges de mouvements et de mouvements d'actes linguistiques»). Ces niveaux hiérarchiques de descriptions sont utilisés dans un certain nombre de systèmes de dialogue dont celui de Cawsey : EDGE. Une transaction est une discussion sur un seul topique (sujet ou concept à expliquer), ouverte par un méta-commentaire sur ledit topique.

Une des limitations de EDGE concerne justement la détermination des séquences d'échanges pouvant être une transaction. D'où la nécessité d'un niveau de contrôle supérieur (comme indiqué dans les spécifications faites plus haut) raisonnant sur le dialogue lui-même.

Dans EDGE, un des opérateurs de dialogue peut (s'il est appliqué) contraindre la transaction à consister en un échange d'ouverture, suivi de quelques échanges sur le sujet et enfin un échange de fermeture. Il planifie ainsi toute la transaction. Cependant, d'autres opérateurs «postent» des sous-buts pour faire connaître un sujet particulier à l'utilisateur. Pour réaliser l'un de ceux-ci, le contrôle est donné aux opérateurs de planification de contenu qui à leur tour peuvent déterminer des buts de dialogue de plus bas niveau pour avoir quelques échanges avec l'utilisateur sur une proposition.
Le processus de planification procède incrémentalement, de telle sorte que le plan n'est pas complètement déterminé lorsque l'explication commence. Initialement, le but principal du dialogue est posté sur un agenda. Lors de l'exécution, le (sous-)but le plus prioritaire est sélectionné et un des opérateurs est choisi pour l'exécuter (chaque opérateur a des contraintes d'application). Les interactions avec l'utilisateur génèrent de nouveaux (sous-)buts. Idéalement, les opérateurs de dialogue comme de détermination de contenu doivent prendre en compte tous les éléments pertinents du contexte (historique de la section, modèle de l'environnement cognitif de l'usager, modèle de l'utilisateur, etc.). Il en est de même du processus d'interprétation de la requête.

Contrôle

Dans un contexte de dialogue ou pas, nous avons vu que les opérateurs sont choisis par des contraintes et des heuristiques. Ces dernières ne peuvent être sûres. Il faut donc pouvoir revenir sur ces choix ou, comme le fait [Lemaire 91], les retarder en gérant plusieurs alternatives en parallèle et en éliminant celles qui deviennent trop lourdes ou moins efficaces que les autres (la «bonne» explication est ainsi très liée au contexte; il est utile de fixer un seuil d'inefficacité maximum décroissant avec le temps de recherche et le nombre d'hypothèses examinées).

Plus globalement, remarquons que chacune des différentes phases de l'explication fait des choix qui détermine le travail de la phase qui la suit : saisie de la requête de l'utilisateur, interprétation

de cette requête, choix d'une stratégie appropriée, construction de l'explication correspondante et transmission de l'explication (ceci n'est qu'un exemple). Lorsqu'au cours d'une phase, un des choix précédemment faits apparait mauvais, le mécanisme de contrôle doit permettre de revenir en arrière ou de corriger directement ce qui doit l'être.

Des techniques d'évaluation de la qualité des explications produites sont donc nécessaires.
Lors de la phase de planification, elles sont liées ou confondues avec les techniques opportunistes corrigeant le contenu des explication déterminé par les techniques de planification. Il est naturel et juste de considérer les «techniques» opprtunistes comme des opérateurs d'amélioration (ajout, modification) ayant des contraintes d'application et concourant à des buts plus ou moins généraux (filtrer, regrouper, déplacer, justifier, etc.).
Tandis que les opérateurs de planification sont en concurrence pour réaliser un certain but, les opérateurs d'amélioration doivent plutôt être déclenchés de façon à ce que la somme de modification soit sinon la plus grande possible, du moins optimum. La gestion de cet optimum et celle de l'évaluation de la qualité de explication en cours fait partie des choix du concepteur.

Conséquences sur l'architecture du module d'explication

Nous avons vu qu'une bonne explication était le résultat de la coopération ou de la collaboration de divers opérateurs (collaboration à cause des opérateurs d'amélioration «dirigés par les données»). Pour contrôler cette coopération/collaboration plusieurs implémentations sont possibles : chaînage arrière, blackboard, etc. Si une architecture distribuée semble mieux prendre en compte le caractère par essence opportuniste des explications, un algorithme ou un moteur d'inférences peut très bien recréer cette liberté. Ceci est donc également du ressort du concepteur.

Dans tous les cas, l'architecture et les opérateurs utilisés doivent être le plus possible génériques pour des raisons d'adaptabilité (à de nouveaux contextes d'explication ainsi qu'à l'ajout de nouvelles stratégies explicatives). Ceci n'a guère pu être approché dans les recherches sur les explications, quoique le système de Cawsey et celui de Moore & Paris soient des étapes dans cette direction.

Pour sa part, [Moore 91] souhaite que son système d'explication puisse un jour, en plus d'être extensible, avoir la capacité de modifier ses stratégies existantes et apprendre de nouvelles stratégies sur la base des interactions avec les utilisateurs.

Chapitre 5 Des modèles d'expertise pour le modèle de coopération

Ce titre exprime que les connaissances acquises aux chapitres précédents vont nous permettre de spécifier plus précisément le modèle de coopération proposé par KADS et ce, dans le langage à quatre niveaux utilisés pour le *Modèle d'Expertise* (sous entendu, de résolution de problèmes; nousn'employons les majuscules que pour ce modèle d'expertise là). Les spécifications que nous proposons sont **méthodologiques** et destinées à guider l'extraction et la modélisation de connaissances lors de la construction d'un SBC devant fournir des des explications adaptées à l'utilisateur, i.e. à un niveau d'abstraction adéquat et en rapport avec les objectifs ou habitudes de celui-ci.

Nous avons vu au chapitre 3 comment la phase d'analyse de KADS aboutit à la création d'un Modèle d'Expertise modélisant au «knowledge level» les connaissances de résolution, et comment la phase de conception conserve les structures de haut niveau de ces connaissances, lors de la création du système final. Les tâches explicatives du SBC peuvent donc retrouver, à partir du code utilisé lors d'une résolution de problème, les connaissances abstraites mises en oeuvre, et ainsi expliquer à un bon niveau d'abstractionles résultats, raisonnements ou connaissances du SBC. C'est l'idée principale du paradigme du «Système Expert Explicatif» [Neches 85] qui fut mise en oeuvre par les générateurs de tels système en conservant l'historique de la transformation du modèle conceptuel vers le SEE final [Neches 85][Paris 92][Kassel 92].

Nous allons donc dans ce chapitre nous proposer une **structure** minimum possible que doit avoir un Modèle Conceptuel afin que le SBC puisse fournir de «bonnes» explications. Nous spécifierons dans ce but et pour guider le cogniticien dans leur acquisition les *types de connaissances* que ce Modèle Conceptuel doit comporter. Nous avons rassemblés les types de connaissances que divers travaux ont montré comme indispensables aux explications [Chandrasekaran 89] [Thomas 92] [Kassel 92] : connaissances de résolution structurées, connaissances explicatives qui leur sont associées, connaissances de stratégies explicatives, modèle utilisateur, historique de la résolution, historique des échanges avec l'utilisateur, etc.

Voyons les insuffisances du Modèle Conceptuel de KADS-I[1] en ce qui concerne la modélisation des expertises liées aux explications. Il est composé d'un *Modèle d'Expertise* (de résolution de problèmes) et d'un *modèle de coopération*. Le premier modélise bien les connaissances de résolution grâce au langage KCML aux quatre niveaux : domaine, inférence, tâche et stratégie.
Par contre, le second n'est pas un modèle d'expertise car il n'inclut que des définitions sommaires des tâches de transferts d'informations entre le système et l'utilisateur. Dans [De Greef 92], les auteurs proposent pour compléter la modélisation des fonctions d'interaction avec l'utilisateur (utilisées dans les tâches de transferts) de construire un ou des «modèle(s) de communication» (cf figure 1). Cependant ces modèles pourraient inclure des détails dépendant de l'inplémentation, comme l'interface avec l'utilisateur, et n'appartiendrait donc pas à la phase d'analyse.
Nous pensons que si l'expertise de communication est suffisante, le modèle de communication devrait être structuré comme le Modèle d'Expertise et que KCML peut être utilisé pour cela. Ce nouveau modèle d'expertise doit ensuite être raffiné durant la phase de conception pour prendre en compte les contraintes d'implémentation.

Enfin, il n'existe pas dans KADS de modèle permettant de formaliser, s'il y a lieu, l'expertise de «coopération avec l'utilisateur», au sens propre du terme, e.g. une répartition interactive du travail [De greef 92].

---

1. Les extentions actuelles de KADS effectués dans le projet KADS-II portent principalement sur le langage de modélisation de l'expertise. Comme nous allons le voir notre extention est donc en fait également appropriée pour KADS-II.

Donc, pour modéliser de façon adéquate les expertises liées à la génération d'explication, i.e. l'expertise de communication et éventuellement celle de coopération, nous proposons de construire à la place du *modèle* (dit) *de coopération*, un «Modèle d'Expertise de Communication» et un «Modèle d'Expertise de Contrôle de la Coopération».
Nous abrévions ces deux nouveaux modèles d'expertise, à construire avec KCML donc, respectivement par **ME-Communication** et **ME-Contrôle**. De même, nous renommons **ME-Résolution**, le *Modèle d'Expertise* de résolution de problèmes.

Notons que l'extension proposée consiste simplement à faire bénéficier d'une «modélisation» d'autres expertises que celle de résolution. Par conséquent :
1) cette extension n'est utile que s'il existe une expertise relative aux explications; si une véritable gestion de la coopération avec l'utilisateur n'est pas envisagée, il suffit de ne pas construire le ME-Contrôle;
2) l'extension est indépendante du langage utilisé pour modéliser l'expertise, et peut donc également s'appliquer au Modèle Conceptuel de KADS-II (comme le souligne Breuker, la modélisation de la coopération et de la communication dans KADS-II est actuellement identique à celle de KADS-I; elle est décrite dans [De Greef 92];
3) nos spécifications sont du **même** niveau d'abtraction que celles de KADS, i.e. **méthodologiques** et indépendantes de toute implémentation.

Nous justifions tout d'abord plus en détail l'introduction de ces deux nouveaux d'expertise en rappelant rapidement comment la coopération et de la communication sont modélisées durant la phase d'analyse de KADS (ceci est extrait ou résumé de [De Greef 92]) et en montrant les insuffisances de cette modélisation pour la constructions de SBC réellement explicatifs ou coopératifs.
Puis pour guider le concepteur dans la construction de notre Modèle Conceptuel étendu, nous listons les différents composants qu'ils doit inclure et détaillons les relations qu'entretiennent ces composants, puis nous proposons des contenu et des modélisations pour les nouveaux modèles d'expertise introduits.

1 Insuffisances de la phase d'analyse

1.1 Rappels sur l'analyse standard de la coopération et de la communication dans KADS

Dans KADS, les connaissances relatives aux explications sont «analysées» lors de la construction du modèle de coopération et du (ou des) modèles de communication. Aussi, rappelons brièvement cette partie du cycle de vie (plus détaillée au chapitre 3).

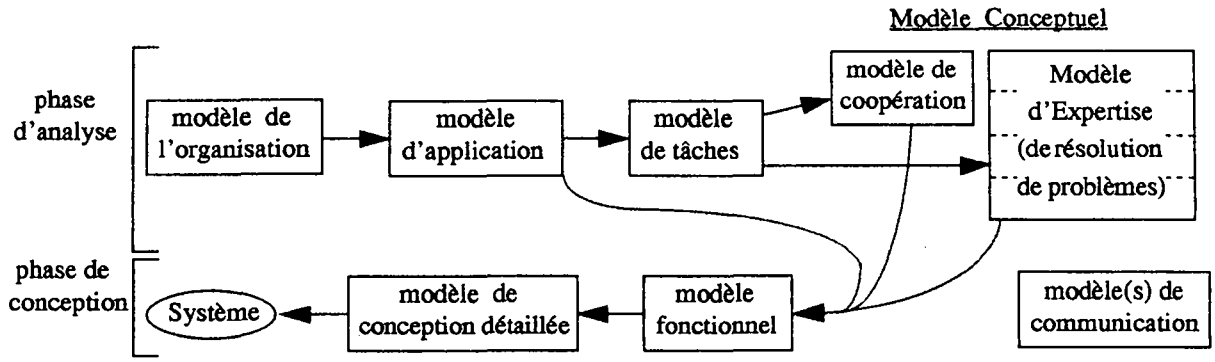


Figure 1: Les classiques modèles du cycle de vie de KADS-I (d'après [Wielinga 92])

- Le *modèle de tâches* fixe les tâches et sous-tâches de la ou des résolution(s) de problèmes, et les distribue entre le système et l'utilisateur (éventuellement suivant le type d'utilisateur). Les échanges entre les tâches du système et celles de l'utilisateur sont attribués à des **tâches de transfert**. KADS propose de spécifier par une **structure de tâche**, l'alternance entre les tâches du système et les tâches de transfert. Remarquons qu'alors, l'interaction entre le système et l'utilisateur, ou encore leur coopération, est prédéfinie et donc assez rigide.

- Le *modèle de coopération* n'est actuellement dans KADS qu'une simple annexe du modèle de tâches, définissant sommairement les tâches de transfert, i.e. les tâches de transmission et de réception entre le système et l'utilisateur : données ou résultats, explications, dialogues, etc. Pour chacune, il précise qui, de l'utilisateur ou du système, a l'**initiative** de l'échange, et quel est le **type d'«ingrédients»** transférés : «information», «connaissance», «talent», etc. (cf De Greef 92].
  Bien sûr, cette modélisation n'est suffisante que pour des échanges très simples : entrée de données, affichage de résultat, etc. C'est pourquoi De Greef et Breuker proposent de construire, pour chaque tâche de transfert ou globalement pour toutes, un ou deux **modèles de communication** traitant respectivement l'«**interprétation de requêtes et la présentation de réponses**» et la «**gestion des dialogues**». Ces modèles seraient plus ou moins élaborés suivant la complexité de la tâche. Dans [De Greef 92] les auteurs ne précisent pas quand construire ces modèles (qui semblent cependant plus appartenir à la phase de conception que celle d'analyse), ni comment les construire, mais cite leur contenu possible : connaissances sur les langages (lexique, syntaxe, sémantique, pragmatique) et sur les règles de dialogue à utiliser dans les transferts, comment utiliser ces connaissances pour satisfaire ou faire comprendre une intention, et enfin comment décider de ces intentions.

- Le *Modèle d'Expertise* rassemble et structure l'expertise de résolution de problèmes. Les connaissances de ce modèle sont d'un bon niveau d'abstraction et nous semblent donc pouvoir être **exploitées par les tâches explicatives** [Chandrasekaran 89] listent les connaissances de résolution nécessaires pour différents types d'explication de résolution de problème). Dans KADS-I, le langage KCML à 4 niveaux est utilisé pour cette modélisation.

- Le *Modèle Conceptuel* est l'ensemble de ces deux derniers modèles plus, selon [De Greef 92], l'ajout aux niveaux tâche et stratégie du Modèle d'Expertise, de la **structure de tâche** (cf. le *modèle de tâches*) de l'interaction entre le système et l'utilisateur.

1.2 Insuffisances de cette analyse pour les expertises liées aux explications

1.2.1 Nécessité d'un modèle d'expertise de coopération

Lorsque la coopération prévue entre le système et l'utilisateur est assez rigide et peut être ainsi relativement bien prédéfinie, l'**ajout** au Modèle d'Expertise de la **structure de tâche** décrivant l'interation système/utilisateur, se justifie par sa simplicité (cf chapitre 3, section 2.2).
Mais lorsque cette coopération est un peu plus dynamique ou compliquée, une structure de tâche est insuffisante : il faut la structure d'un modèle d'expertise pour modéliser les concepts, règles, tâches et stratégies de la coopération. En effet, une rélle gestion de la coopération, e.g. la répartition interactive du travail entre le système et l'utilisateur, implique des raisonnement complexes car elle dépend de nombreux **facteurs** : des connaissances, des aptitudes et des buts de l'utilisateur, des diverses tâches que le système doit accomplir ou bien faire accomplir à l'utilisateur par exemple pour le faire progresser ou bien obtenir des informations. De plus, l'évolution de l'utilisateur et des priorités entre les divers objectifs à atteindre étant susceptibles d'évoluer, la coopération peut devoir être gérée dynamiquement, en fonction de ces divers paramètres ou facteurs.
La coopération est un aspect important des explications puisqu'elle contrôle à un haut niveau les interactions entre le système et l'utilisateur. Par exemple, le système peut introduire de nouveaux sujets à aborder dans les dialogues. Si la coopération est dynamique, elle induit une négociation du travail à accomplir avec l'utilisateur (e.g. [Hewit 91][Gasser 91][Brézillon 92]). On peut rencontrer de telles collaborations entre l'homme et la machine dans certains systèmes d'enseignement ou de mises à jour de bases de connaissances.

En résumé, pour des systèmes coopératifs importants, les connaissanses relatives à la coopération forment une expertise à part entière, distincte de celle de la résolution de problèmes et qui doit donc être **modélisée dans un modèle d'expertise différent**. Il ne serait en effet pas très modulaire et pratique de mélanger dans un seul modèle d'expertise, des connaissances ayant des rôles aussi différents : résolution vs coopération.
Ce nouveau modèle peut être structuré avec KCML, le langage de modélisation d'expertise de KADS-I. Nous le nommons **ME-Contrôle** car, comme nous allons le voir, il gère afin d'atteindre les buts coopératifs déterminés, à la fois les tâches de résolution et celles de communication (voir figure 2).

1.2.2 Nécessité d'un modèle d'expertise de communication

Pour expliquer les connaissances et surtout les solutions fournies par un SBC, les raisonnements explicatifs nécessaires sont complexes et multiples : synthétiser le «raisonnement» de résolution, exposer la solution ou critiquer une solution alternative, justifier les choix, s'adapter aux connaissances, habitudes et évolutions de l'utilisateur, gérer un dialogue avec lui pour répondre au mieux à ses attentes, etc.
Or le(s) *modèle(s) de communication* de De Greef et Breuker appartiennent essentiellement à la phase de conception. Les raisonnements explicatifs ne sont donc pas préalablement modélisés comme ils devraient l'être. Nous suggérons donc d'inclure au Modèle Conceptuel, un «Modèle d'Expertise de Communication (ME-Communication). En effet, comme pour l'expertise de coopération, nous ne conseillons pas de mélanger l'expertise de Communication avec celle de Résolution (de manière analogue, [Brézillon 92] et [Thomas 92] préconisent de constituer deux SBC séparés). Il est cependant important de comprendre que les tâches de communication utilisent comme **des données** les connaissances du ME-Résolution (par exemple pour expliquer un concept ou un résultat de résolution).

Avant de détailler nos deux nouveaux modèles, la section suivante précise les différentes relations qu'ils entretiennent et les données qu'ils utilisent.

2 *Une nouvelle architecture*

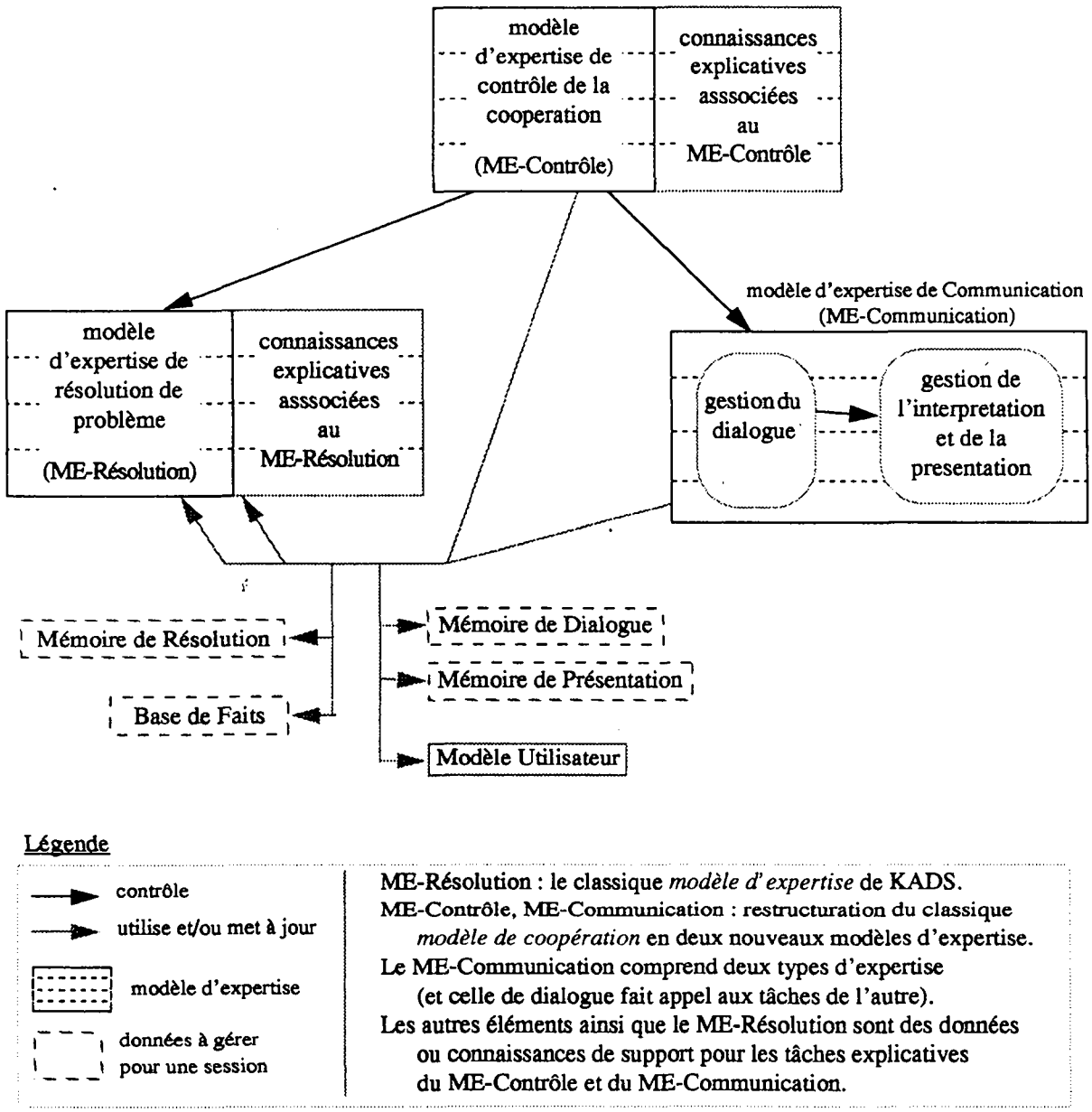


Figure 2: Proposition de sous-modèles d'expertise pour le Modèle Conceptuel

Cette figure montre comment les trois modèles d'expertise du Modèle Conceptuel interagissent et quelles données ils utilisent. **Précisons certains points de cette architecture.**

Le ME-Contrôle détermine les buts de résolution et de communication à atteindre pour gérer la cooperation avec l'utilisateur puis déclenche les tâches de résolution et de communication afin d'atteindre ces buts.
Les tâches du ME-Contrôle et du ME-Communication travaillent sur différents types de «données et connaissances de support pour les explications». Nous détaillerons chacun de ces types dans la section 1.6. Le ME-Résolution fait partie de ces connaissances de support pour les explications puisqu'il contient des connaissances de résolution de problème à un haut niveau d'abstraction : concepts, relations entre concepts, inférences, tâches, stratégies, etc.

Mais pour de bonnes explications, les descriptions du ME-Résolution doivent être commentées ou complétées par une méta-connaissance : justifications, définitions, schéma explicatif, figures, etc. Nous appellerons l'ensemble de cette méta-connaissance «connaissances explicatives associées au ME-Résolution».
Par analogie, si les tâches de cooperation doivent être expliquées, des «connaissances explicatives associées au ME-Contrôle» doivent être extraites.
Des Modèles des Utilisateurs sont essentiels pour des explications souples et ciblées (idem pour une bonne coopération). Ils doivent donc être acquis.
Enfin, les tâches explicatives et coopératives travaillent sur les **traces** de résolution de problèmes, les données du problème, et les précédents échanges d'information avec l'utilisateur. Dans la figure 2, ces éléments appartiennent respectivement à la «Mémoire de Résolution», la «Base de Faits» et la «Mémoire de Dialogue». Ce sont des «données de support pour les explications» (cf section 1.6.). Ces données sont décrites par des concepts des niveaux domaine du ME-Contrôle et du ME-Communication et sont référés par des méta-classes aux niveaux inference et tâche de ces modèles. Nous avons fait figurer ces «données de support pour les explications» dans la figure 2 pour montrer quels éléments doivent être exploitées par les tâches explicatives et coopératives, en même temps que les «connaissances de support pour les explications».

Notons que KADS n'interdit pas de construire plusieurs «modèles d'expertise de résolution de problèmes» pour modéliser différents types d'expertise de résolution de problèmes intervenant dans une application. Notre architecture s'adapte aisément à cette exigence de modularité si le cogiticien le désire : il lui suffit de construire plusieurs «ME-résolution et connaissances explicatives associées». Le contenu des autres modèles d'expertise n'ont pas à être modifiés puisque leur tâches exploitent en tant que **données** les connaissances des ME-Résolution. Mais le ME-Contrôle **disposera** de plusieurs expertises de résolution pour répondre aux besoins de l'utilisateur.

Ainsi, la *modularité* de cette architecture et le langage de modélisation d'expertise de KADS permettent d'extraire et de structurer convenablement les différentes expertises nécessaires.
De plus, la plupart des techniques s'appliquant au ME-Résolution devraient aussi pouvoir être appliquées aux nouveaux modèles d'expertise, par exemple :
- les techniques exploitant un modèle d'expertise pour les explications ([Baumewerd 91][Sprenger 92];
- les techniques d'extraction, de conception et d'implémentation.

L'acquisition des expertises de coopération et de communication est ainsi facilitée.

Nous allons maintenant préciser les éléments de cette architecture et ensuite la connaissance explicative qui doit être extraite. Comme l'expert en résolution de problèmes n'est pas un expert en explication, le cogniticien devra définir lui-même la plupart des concepts et stratégies explicatifs. Notons en effet que :
- les tâches explicatives humaines ne peuvent être connues qu'à un niveau d'abstraction assez élevé mais ceci n'est pas un gros inconvénient étant donné la différence de fonctionnement entre un ordinateur et un cerveau humain ([Lemaire 91a]).
- le cogniticien construit le ME-Contrôle et le ME-Communication guidé par les fonctionnalité attendues en coperation et en explication de l'application.

Il est donc absolument correct de spécifier l'exploitation de la «Mémoire de Résolution» dans le ME-Communication. Rappelons que durant la phase d'analyse de KADS, modéliser ou définir un élément *peut ne consister qu'à mentionner son existence*. Cet élément sera ensuite raffiné durant la phase de conception. Le cogniticien est donc responsable de la précision des spécifications de l'analyse. Il ne peut de toute façon spécifier durant la phase d'analyse toutes les stratégies explicatives envisagées puisque certaines dépendent de choix de conception, par exemple de l'interface système/utilisateur.

2.1 Données et connaissances de support pour les explications

Il s'agit de données sur lesquelles travaillent les tâches du ME-communication et du ME-Contrôle. Elles peuvent donc être référées par des méta-classes dans les structures d'inférences des niveaux tâche de ces modèles. L'utilisation de ces connaissances a été montré dans les chapitres 1 et 4 et sera (re-)précisé plus loin.
La liste que nous dressons ci-dessous de ces connaissances n'est pas exhaustive mais tous ses éléments sont clairement indispensables pour générer de bonnes explications. Mieux ils sont **exploités**, plus les explications pourront être complètes ou adaptées à l'interlocuteur. Les méchanismes de ces **exploitations** doivent donc être acquis ou définis en fonction des besoins en explications. Durant la phase de conception, il faut de plus prendre en compte le temps nécessaire au système pour exploiter chacun de ces éléments.

- La «**Mémoire de Résolution**» : il s'agit d'une trace organisée des contextes de résolutions passées ou en cours : buts poursuivis, tâches et méthodes activées, etc.
  Dans le but de fournir de bonnes explications, les traces du système expert SMACK [Thomas 92] enregistre toutes les décisions prises par le système à un moment donné (avec justifications et conséquences), leur valeur de vérité à l'instant présent, et les dépendances entre décisions sous la forme de combinaisons et/ou.
  Les éléments de cette trace sont liés au code du système final. Ils ne sont donc généralement pas d'un niveau d'abstraction suffisant pour servir directement aux tâches explicatives (cf chapitre 1). Aussi, comme indiqué dans l'introduction de ce chapitre, un mécanisme doit permettre de retrouver à partir de ces éléments, les connaissances du ME-Résolution qu'ils font intervenir : stratégies, tâches, buts, structures de tâches, inférences, relations, méthodes, etc.

- La «**Base de Faits**» : ce sont les données relatives au(x) cas traité(s) par le ME-Résolution lors d'une exécution ou d'une session; celles d'entrée (e.g. dossier médical et symptômes pour un diagnostic) et celles déduites (e.g. rougeole).
  Ces données doivent être reliées, par un mécanisme d'abstraction (comme pour la Mémoire de Résolution) à des «méta-classes» du ME-Résolution (notamment celles de la structure d'inférence en cours d'exécution).

- Le **ME-Résolution** : comme il contient les connaissances du domaine, les rôles qu'elles peuvent jouer, les inférences, tâches et stratégies utilisables, il est indispensable pour fournir des explications et comprendre les requêtes de l'utilisateur. Le mécanisme d'abstraction de traces ou de données cité ci-dessus réfère à des éléments du ME-Résolution; pour les exploiter, il faut connaître leurs relations avec les autres éléments du ME-Résolution. Cette exploitation est developpée dans la sous-section suivante..

- **Les connaissances explicatives associées au ME-Résolution** : elles peuvent être structurées comme un modèle d'expertise. Pour les décrire, supposons qu'elles soient structurées avec KCML. Le *niveau domaine* contient alors essentiellement les connaissances profondes (e.g. réseau causal justificatif, etc.) et/ou explicatives (e.g. définition, justification, schéma explicatif, figures, etc.) qui sont **liées** aux éléments du niveau domaine du ME-Résolution mais qui ne servent pas à la résolution de problèmes.
  Ce niveau domaine peut également contenir des «interprétations en vue de l'explication» des éléments des autres niveaux du ME-Résolution. Ces «interprétations», qui peuvent consister en des schémas explicatifs, sont exprimées dans les concepts du niveau domaine du ME-Résolution. Ainsi, les inférences, tâches et stratégies peuvent être expliquées dans le langage du domaine de l'expertise.
  Les *autres niveaux* de ce modèle d'expertise sont logiquement indépendants du domaine de l'expertise. Ils permettent de contenir des schémas d'explications des modèles d'interpré-

tation utilisés dans le ME-Résolution. Le recueuil de ce type de schémas est au centre de nombreuses recherches (e.g. [Safar 92]).
Si elles sont structurées comme un modèle d'expertise, ces connaissances explicatives peuvent éventuellement être exploitées par les mêmes algorithmes que les connaissances de résolution auquelles elles sont associées.
Ces descriptions s'appliquent également aux «connaissances explicatives associées au ME-Contrôle» lorsqu'elles sont recueuillies.

- Le «**Modèle de l'Utilisateur**» (MU) : construit par avance et/ou adapté et mis à jour lors de chaque interaction avec l'utilisateur, c'est un modèle de ses : buts, centres d'intérêts, habitudes, méthodes, types de questions, connaissances sur le domaine, types de difficultés et/ou facilités d'apprentissage ou de compréhension, préférences, connaissances sur le modèle utilisé par le SBC (le Modèle Conceptuel dans notre cas), ..., ainsi que, au cours d'une session, son modèle d'environnement cognitif (cf fin de la section 2 du chapitre 4). A moins d'être demandés explicitement, les éléments de cette liste nécessitent des processus très complexes pour être acquis (cf [Maybury 90]). Cependant, même grossiers, ces éléments aident à générer des explications flexibles et bien adaptées à l'interlocuteur. Il peut y avoir autant de MU que d'utilisateurs ou bien autant que de classes d'utilisateurs, e.g. «débutants» vs «experts».

- La «**Mémoire de Présentation**» et la «**Mémoire de Dialogue**» : ce sont les pendants de la «Mémoire de Résolution» pour les expertises du ME-Communication. Comme ces historiques de génération sont structurés et abstraits (par un mécanisme d'abstraction comme décrit plus haut), ils doivent permettre d'**associer** aux explications produites, les divers buts poursuivis lors de leur génération et les divers choix effectués.
Nous avons vu lors de l'étude des mécanisme de planifications que ceci était indispensable pour assurer un dialogue ou répondre à une question sur une explication précédemment fournie. Ces mémoires permettent également aux tâches de dialogue et de présentation de raisonner sur les dialogues pour gérer les sujets abordés et à aborder.

Observons maintenant plus précisément **comment l'exploitation des niveaux KADS du ME-Résolution peut fournir une grande part du matériel explicatif.**

2.1.1 Exploitation du ME-Résolution pour les explications

Selon les objectifs de la tâche d'explication, d'après la typologie proposée par [Chandrasekaran 89] et détaillée au chapitre 1, diverses connaissances sont impliquées.

- **Pour justifier ou présenter les connaissances elles-mêmes (type 2) :** *tout le ME-Résolution et ses méta-connaissances explicatives associées.* Toutes ces connaissances étant explicites et séparées par niveau, il est possible de présenter : 1) un concept du domaine en le situant dans sa taxonomie, en montrant ses propriétés et ses liens avec d'autres concepts; 2) les différents rôles de ces concepts et à quel endroit du raisonnement; 3) une tâche avec ses conditions, son contrôle et ses décompositions possibles (et ceci peut être utilisé en mode trace); 4) les diverses stratégies.

- **Pour justifier le lien entre données et résultats (type 1) :** *domaine du ME-Résolution, Base de Faits et Mémoire de Résolution.* Il n'est pas utile de présenter les méthodes mises en oeuvre pour trouver le résultat (seule la chaîne de causalité i.e. la trace compte; exemple de question : «Pourquoi le patient est-il atteint de telle maladie ?»).

- **Pour justifier la méthode de résolution mise en oeuvre (type 3)** pour résoudre un problème précis : *niveaux stratégie et tâche du ME-Résolution* (exemple de question : «Comment le système a-t-il conclu à telle maladie ?»).

Opérons avec [Baumewerd 91] le travail inverse et observons à quels types de question peuvent répondre chaque élément du ME-Résolution.
Ainsi, dans les requêtes de l'utilisateur peuvent être *recherchés les éléments du ME-Résolution concernés*. Puis, des liens que chacun de ces éléments (relation, concept, méta-classe, source de connaissance (SC), tâche, stratégie) entretient avec d'autres éléments du ME-Résolution, **déduire** *les connaissances qu'il manque à l'utilisateur pour qu'il ait eu à poser cette question.*
Si un processus de résolution est en cours, les questions concernent implicitement l'étape courante (mais l'usager peut explicitement désigner n'importe quelle étape).

- Le niveau domaine permet de répondre aux questions sur la connaissance elle-même. Ex: «quelles sont les propriétés du plomb ?» ou «quelle est la concentration maximum autorisée de plomb dans l'eau potable ?». L'utilisation d'une base de données dans l'implantation du niveau domaine permet de répondre à ce type de question.
- Le niveau inférence permet de répondre à des questions comme les suivantes (le mot «rôle» y a sons sens commun plus celui de méta-classe) :
  «pourquoi l'évaluation de la qualité du sol est nécessaire ?» du type «**conséquences de tel rôle**» (*recherche des inférences* (utilisées ou en cours) *dont une méta-classe d'entrée réfère à l'élément de la question*),
  «pourquoi la valeur de l'impact sur le sol a été jugée 'élevée' ?» du type «**ce qui conduit a tel rôle**» (*même recherche, mais sur les rôles de sortie*) ,
  «comment la valeur de l'impact sur le sol est-elle déterminée ?» du type «**entrées de la SC ayant conduit au rôle**» ou du type «**bloc d'inférences ayant conduit au rôle**»,
  «comment la contamination du sol par le plomb est-elle échelonnée (classifiée)» ou «quelles autres méthodes existent» du type «**justifier telle SC**» (*par référence aux relations ou méthodes qu'elle utilise*).
- Le niveau tâche permet de répondre à des demandes d'informations sur le contrôle : type «**pourquoi utiliser telle SC**», type «**comment telle tâche**», type «**intérêt de telle tâche**».
- Le niveau stratégie permet de répondre à des questions du type «**pourquoi telle tâche**».

D'autres types de questions n'appartiennent pas à un niveau précis. Il s'agit :
- du type «**pourquoi pas telle tâche**» et du type «**pourquoi pas telle SC**»;
  exemple : «pourquoi n'avoir pas recherché l'impact du plomb dans l'air?»; la circonstance ayant empêché cette SC de s'exécuter peut être sémantique ou due à des contraintes du domaine, au contrôle de la tache, etc.;
- du type «**qu'est-ce qui se passerait dans telle situation**»;
  exemple : «quelles seraient les conséquences si la concentration de plomb dans l'air était de 0,5% au lieu de 1% ?». Comme les questions peuvent être posées au cours du raisonnement, celui-ci devra alors ne pas être monotone.

Dans [Sprenger 92], l'auteur approfondi ce travail en donnant des algorithmes exploitant différemment les différentes couches du modèle, suivant le but explicatif poursuivi.
Ce *principe de recherche* est d'une grande aide tant pour l'**interprétation** des requêtes que pour la **recherche des réponses** : il fournit le *matériel initial* et permet de générer des explications dans le *langage* du Modèle Conceptuel. Cependant, si l'utilisateur ne tient pas à connaître l'expertise tel qu'elle est décrite dans le ME-Résolution, les liens entre ses questions et les éléments du ME-Résolution seront plus difficile à trouver et d'autre part, il faudra *construire* (tout de même en s'aidant du matériel acquis par ce principe) une explication qui soit assez proche du modèle de résolution de l'utilisateur (et donc assez *éloignée* du ME-Résolution).

Si ce type d'exploitation fait partie de manière évidente de l'arsenal de base des tâches de Présentation (interprétation, génération), celles de dialogue ont aussi besoin de bien utiliser ou comprendre le ME-Résolution pour leurs différentes gestions.

3 Spécifications des nouveaux modèles

3.1 Le Modèle d'Expertise de Contrôle de la coopération

Pour les systèmes coopératifs importants, le ME-Contrôle doit contenir les concepts, règles, méthodes, tâches et stratégies de partage de travail, de négociation de ce partage avec l'utilisateur, et de contrôle à un haut niveau d'autres interactions avec celui-ci, comme les échanges d'informations, les explications.

Pour le partage de travail, il doit gérer de nombreux **impératifs** (adaptation aux connaissances et aux attentes de chacun des utilisateurs, tâches à accomplir ou faire accomplir, priorités entre les utilisateurs et entre les objectifs, ...) et **ressources** (plusieurs utilisateurs, plusieurs expertises, ...).

Les techniques actuelles de gestion de dialogue et de génération de discours travaillent **en local sur des buts de discours**. Pour modéliser les tâches de transfert, de telles techniques doivent être décrites dans le Modèle de Communication. Les tâches de ce dernier sont donc utilisées et dirigées par le ME-Contrôle, lors des interactions entre le système et l'utilisateur :
- en **déterminant** en fonction des mêmes impératifs que ci-dessus, les «**messages**» à faire passer auprès de l'utilisateur et en **fixant** plus ou moins les **sujets ou concepts** à aborder et développer;
- en **déterminant** «quand changer» ou «quand a changé» l'**objectif** central d'une transaction en cours avec l'utilisateur. Ceci nécessite une gestion séparée de celle du dialogue [Cawsey 91a] et relève assez naturellement des tâches à affecter au ME-Contrôle.

Ainsi, les explications proprement dites sont gérées par les tâches du ME-Communication bien que le ME-Contrôle puisse influencer le choix de tel ou tel concept. De plus, comme les aspects coopératifs du dialogue doivent logiquement aussi être modélisés dans le ME-Communication, le ME-Contrôle est très petit pour des applications courantes (faiblement coopératives).
Quand les explications n'ont pas besoin d'être très souples ou adaptées à l'utilisateur, le ME-Communication est lui aussi réduit. Mais pour des raisons de modularité, il peut sembler bon d'isoler ces quelques connaissances de génération d'explication de celles de résolution de problèmes.

Supposons que le concepteur décide de construire un ME-Contrôle pour modéliser une collaboration très souple ou dynamique, de quelles sources d'expertise dispose-t-il ?

- Tout d'abord, celle de l'expert du domaine ou d'éventuels experts «en collaboration». Nous avons vu avec les explications que cette «expertise» était répartie dans de nombreuses sciences et difficilement formalisable. L'acquisition devra donc s'appuyer sur le bon sens et sur des expériences comme celles du Magicien d'Oz (où l'expert joue le rôle du futur SBC en communiquant avec l'utilisateur via un terminal). [Lemaire 91a] et [Safar 92] détaillent la façon de mettre en oeuvre et d'exploiter de telles expériences pour obtenir des stratégies d'explication auprès de l'expert.
- De plus, le concepteur peut trouver de bons concepts et méthodes dans l'Intelligence Artificielle Distribuée, qui s'intéresse à la modélisation de la coopération entre agents informatiques : comment des agents autonomes proposent ou négocient une décomposition de tâches ou une distribution (cf par exemple [Hewitt 91] [Gasser 91]). Le chapitre précédent fournit des spécifications (notamment la vision de [Baker 92]) qui pourront aider le concepteur dans ses choix.
- Enfin, lorqu'un tel travail aura été fait, le MCC résultant pourra servir de base pour d'autres applications aussi ambitieuses (car une bonne part d'abstraction aura due être réalisée).

3.2 Le Modèle d'Expertise de Communication

Le ME-Communication comprend l'expertise d'Interprétation/Présentation et l'expertise de Dialogue (les tâches de dialogue utilisant bien sûr celles d'interprétation et de présentation).

Une part de l'expertise de communication qui doit être modélisée concerne donc l'*interprétation* des données fournies par l'utilisateur et *la présentation d'information ou de résultats* à l'utilisateur. Les *données* en entrée peuvent être une *requête* d'information ou d'explications.
Ces deux fonctions représentent les étapes importantes des deux grandes classes de tâches de transfert : celles de réception de données et celles de transmission de données. Les autres étapes relèvent de la phase de conception : collecte ou sortie des données sur l'interface choisi, échange des données entre les tâches de transfert et les tâches qui les utilisent.
Nous montrons dans les deux sections suivantes comment chacune de ces fonctions peut être modélisée dans KADS, notamment en donnant des structures d'inférences, i.e. en quelque sorte des modèle d'interprétation. Voici en guise d'introduction un apercu général :

- L'*interprétation* consiste à produire, à partir de données exprimées dans un langage compréhensible par l'utilisateur (e.g. menus, mot-clés, langage de requête, langage naturel), une *structure de données* dont le format est compréhensible par la machine.
  Les **niveaux domaine et inférence** du ME-Communication doivent donc inclure différents *concepts et méthodes* relatifs :
  1) aux langages de l'utilisateur et du système (pour chacun, le niveau domaine doit contenir son lexique, sa syntaxe, sa sémantique et sa pragmatique)[1]; notons que les notions de médium séquentiel (texte, parole, etc.) et spatial (figure, etc.) permettent de rester assez indépendant de l'interface utilisateur finale;
  2) aux transformations d'un langage vers un autre et l'utilisation des «connaissances de support» pour cela;
  Au niveau **tâche**, le cogniticien définit le contrôle de l'enchaînement de ces méthodes (ou plus précisément des «inférences» réalisées par ces méthodes) en fonction de la souplesse attendue dans la transformation, de l'intéractivité avec l'utilisateur, etc.
- La Présentation consiste à fournir des informations sur des objects internes au système, dans un langage compréhensible par l'utilisateur. Ce langage doit donc lui aussi être défini et les connaissances pour générer du discours avec ce langage doivent être acquises.

L'expertise de Dialogue est moins aisée à définir que celle de Présentation car elle dépend plus de l'application : type de coopération, type de dialogue choisi, etc. De plus, l'état de l'art actuel sur les dialogues ne nous permet guère de donner des structures d'inférences comme pour la Présentation. Cependant, certaines techniques simples de gestion de génération de discours peuvent également s'appliquer aux dialogues [Cawsey 91a] et nous montrons dans la section suivante, pour la présentation, comment ces techniques peuvent être modélisés avec KCML. Voici donc une description générale. Dans un dialogue, chacun des partenaires peut prendre l'initiative d'introduire de nouveaux «topiques» (i.e. contenu ou sujet), modifiant ainsi le type et la direction du flot de données de la tâche de transfert. Selon [De Greef 92], des *protocoles de «prise de parole»*, e.g. via une interruption du flot de données, doivent donc être modélisés, et le *rôle des topiques introduits* déterminés : «nouveaux topiques» ou «sous-topiques» (i.e. reliés à d'ancien topiques).

---

1. - un lexique fait correspondre des objets et actions internes avec des objets et évènements physiques;
   - une syntaxe décrit quel agencement ou séquences temporelles de (types de) entrées lexicales peuvent avoir une interprétation; elle peut permettre d'emboîter des expressions pour en produire de plus complexes;
   - une sémantique donne le sens d'une expression, lequel dépend des représentations internes;
   - une pragmatique décrit comment en pratique utiliser les éléments ci-dessus, e.g. pour tel type utilisateur ou lors d'un dialogue, etc.

Les niveaux KADS du ME-Communication devront donc contenir pour l'expertise de Dialogue (chaque élément de cette liste peut être plus ou moins développé suivant les besoins en explications) :

- **au niveau domaine** : un *modèle* de dialogue (conventions, règles de construction et de coopération) et tous les autres concepts nécessaires aux niveaux tâche et inférence.
- **aux niveaux inférence, tâche, et stratégie** : les divers mécanismes de *gestion* des topiques, de «prise de parole», de modélisation de l'utilisateur (cf section suivante) et d'application d'objectif (i.e. quels sous-topiques ou nouveaux topiques introduire pour satisfaire au mieux l'objectif fixé, en fonction des connaissances de l'utilisateur et de ses attentes; l'objectif pouvant être donné par le ME-Contrôle ou être issu d'une requête de l'utilisateur).
  Ainsi sont inclus : 1) les *explications spontanées* (i.e. données pour prévenir des questions prévisibles de l'utilisateur), 2) l'*adaptation* aux nouveaux buts et connaissances de l'utilisateur, 3) le *maintien et l'exploitation* du contexte du dialogue (sujets précédemment abordés, requêtes, explications, etc.).

Notons que les tâches de dialogue fournissent des topiques aux tâches de présentation. Donc pour être utiles aux tâches de dialogue, les tâches de présentation doivent supporter les interruptions et être capable de poursuivre ou d'améliorer une explication.
La gestion des interruptions et des questions «follow-up» (i.e. sur les explications précédentes) appartient à la fois à l'expertise de Présentation et à celle de Dialogue. La collaboration entre les tâches de ces deux types d'expertise est du ressort du concepteur. De même, durant la phase d'analyse, la modélisation peut être plus ou moins précise. Par exemple, la gestion des interruptions peut n'être modélisée que par une simple source de connaissances, et la description fine des méthodes de cette dernière est ainsi réalisée en phase de conception. Ainsi durant cette phase, le ME-Communication est assez indépendant de l'interface utilisateur finale. La phase de conception permet de prendre en compte les influences des décisions de conceptions sur les explications, et donc de compléter ou de raffiner le ME-Communication.

Pour aider le cogniticien à extraire et organiser les connaissances de résolution en vue de la construction du ME-Résolution, KADS propose une bibliothèque de modèles d'interprétation, i.e. essentiellement des structures d'inférences commentées (cf [Breuker (ed) 87]).
De même nous proposons dans les sections suivantes de telles structures d'inférences pour aider à modéliser les fonctions d'Interprétation et de Présentation. Les sources de connaissances des structures d'inférence sont désormais notées SC.

3.3 Modélisation de l'Interprétation

L'*interprétation* consiste à produire, à partir de données exprimées dans un langage compréhensible par l'utilisateur, une *structure de données* dont le format est compréhensible par la machine. Cette transformation peut être triviale (e.g. transformation d'un chiffre en son code binaire; association d'une commande de menu à la procédure qui l'exécute), de difficulté moyenne (e.g. requête exprimée dans un langage de requêtes) voire très complexe (e.g. interprétation d'une phrase en langage naturel).
Dans les cas complexes, l'interprétation (SC «***transformer***/étendre/sélectionner» dans la figure ci-dessous) s'*accompagne d'une recherche des références implicites* dans les données («***étendre***»), et d'une suppression des éléments inutiles ou redondants («***sélectionner***»). Puis, la structure de donnée ainsi obtenue, est éventuellement ***retransformée*** en un langage compréhensible de l'utilisateur pour lui faire part de l'interprétation qui a été faite de ses données. D'où la structure d'inférence suivante, utilisant les termes conseillés par KADS :

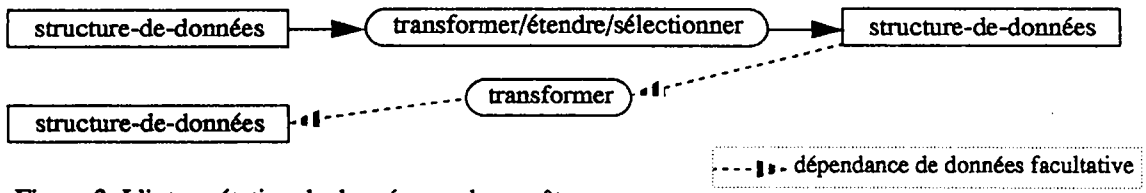


Figure 2: L'interprétation de données ou de requête

Lorsqu'il suit cette structure d'inférence, le cogniticien doit définir ou acquérir pour les niveaux **domaine et inférence** de son ME-Communication, les différents *concepts et méthodes* relatifs
1) aux *divers* langages utilisés : celui permettant à l'utilisateur d'exprimer sa requête, celui qui en représente le sens en machine et, s'il est différent du premier, celui nécessaire pour communiquer à l'utilisateur l'interprétation de sa requête (pour chaque langage, il faut définir son lexique, sa syntaxe et éventuellement sa sémantique et sa pragmatique);
2) aux transformations d'un langage vers un autre;
3) à l'*utilisation* des «connaissances de support», e.g. l'historique du dialogue et le MU pour la recherche des éléments implicites. Notez que les éléments du ME-Résolution sont à la base des lexiques des langages utilisés. La mise en correspondance des éléments de la requête avec ceux du ME-Résolution participe à la fois à l'interprétation de cette requête et à l'apport d'éléments de réponses [Baumewerd 91] [Sprenger 92];
4) à la mise à jour de quelques connaissances de support : historique de Présentation et éventuellement MU et historique de Dialogue.

Au niveau **tâche**, le cogniticien définit le contrôle de l'enchaînement de ces méthodes en fonction de la souplesse attendue dans la transformation, de l'interactivité avec l'utilisateur, etc.

3.4 Modélisation de la Présentation

Le but de la Présentation est ici de fournir dans un langage compréhensible par l'utilisateur, des *informations* sur des objets du système, généralement ceux du ME-Résolution, e.g. connaissances ou résultat de raisonnement. La structure d'inférence que nous proposons ci-dessous (figure 3), traite le cas le plus complexe, celui de la génération d'explications. Elle a donc pour entrée une demande d'explication sur un objet connu par le système. S'il s'agit d'une explication spontanée, cette requête est issue d'une tâche du système, sinon elle est la formalisation d'une requête de l'utilisateur (cf figure 2).
L'objet à expliquer est soit fourni également, soit généré ou recherché à partir de la requête : dans la figure 3, SC «***résoudre***» si une résolution est nécessaire, SC «***sélectionner***» si l'objet est une connaissance d'un modèle d'expertise.

A partir de ces données, la tâche de présentation consiste à rechercher des éléments de réponse dans le modèle d'expertise concerné par la requête (donc généralement dans le ME-Résolution et les descriptions explicatives associées à ses éléments) puis à les assembler pour créer une **structure intentionnelle** combinant les effets que chaque «expression» doit avoir sur l'interlocuteur. Globalement, toutes les techniques actuelles de génération de discours travaillent ainsi pour atteindre des **buts du discours**.
Certaines techniques opèrent une *pré-sélection* de contraintes ou de connaissances à appliquer pour la sélection des éléments de réponse. Ceci est détaillé dans [Suthers 91] sous le nom de «détermination de la perspective» (choix des concepts, propriétés et relations qui serviront de base à l'explication). Cette *pré-sélection* peut s'effectuer avant la *sélection* ou au cours de celle-ci (alternance des étapes). Comme la sélection et l'assemblage peuvent également être alternés, ces inférences ont été *réunies* dans une seule SC («sélectionner/assembler»). Nous verrons plus loin une structure d'inférence plus précise, notamment sur cette «sélection».
La *structure intentionnelle* doit ensuite être *raffinée* (SC «transformer/étendre/raffiner») avant d'être donnée en entrée à un générateur de «phrases» qui la *transforme* pour que son «transfert» sur le support physique soit compréhensible et naturel pour l'usager.

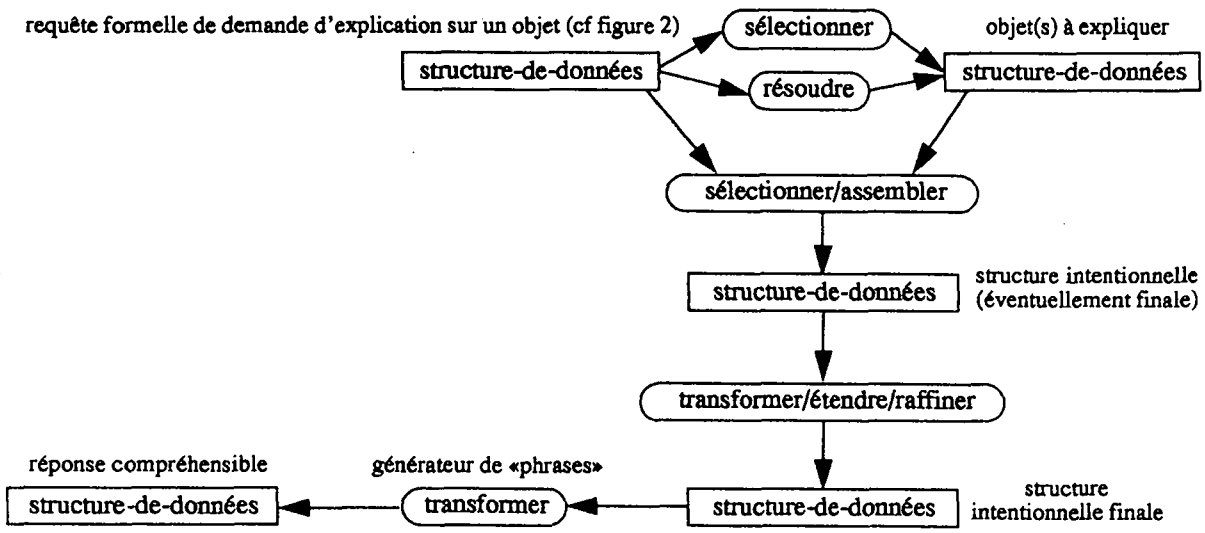


Figure 3: Structure d'inférence générale de la génération d'explications

Au cours de chacune de ces trois grandes phases, tous les éléments des *connaissances de support* que nous avons vues, peuvent être exploités. Ils peuvent également être mis à jour, exceptés bien sûr ceux du ME-Résolution.

Ces **trois phases** sont nécessaires. La dernière est utile, pour des raisons de simplicité et d'efficacité puisque les techniques de planification de discours, qui créent la structure intentionnelle, ne doivent pas gérer en plus les niveaux de détails inférieurs : lexique, grammaire, choix d'expressions ou de tournures de phrase adaptés à l'utilisateur. Ceci est le rôle du ***générateur de «phrases»***, lequel peut également afficher des graphiques (ou des images, etc.) si *le langage destiné à l'utilisateur* admet l'inclusion ou la gestion de ces graphiques et si donc, ils sont inclus dans la structure intentionnelle.
De plus, la structure intentionnelle ne peut être générée d'un seul coup ([Lemaire 92], [Suthers 91]). Après le *mécanisme de planification descendante* généralement utilisé pour déterminer cette structure, des *mécanismes dirigés par les données* sont nécessaires pour la corriger et l'étendre. Ceci, en fonction de *règles pédagogiques*, des *connaissances* de l'utilisateur ou de ses *préférences*, etc.

Pour élaborer cette description et ainsi la modélisation des tâches explicatives, nous devons introduire la notion «d'opérateurs». Tous les mécanismes actuels de création et de raffinement de cette structure font appel à des «**opérateurs**» : sortes de règles avec des conditions de déclenchement, pouvant s'appeler entre elles, et dirigées par les buts ou par les données. Par exemple, des opérateurs dirigés par les données peuvent être sensibles au type des éléments de la structure intentionnelle ou à leur agencement.

> Voici à titre d'exemple, une paraphrase du code de l'opérateur «persuader» de [Moore et Paris 91]. Les mots clefs et les noms d'opérateurs sont en italique.
>
> Pour qu'un système s puisse *persuader* un utilisateur u d'accepter pour *but* de *réaliser* un acte a, il doit vérifier que a est une *étape* dans l'accomplissement d'au moins un *but* commun à s et à u puis *motiver* a par chacun des buts communs à s et à u.

Les opérateurs permettent d'obtenir l'adaptabilité des processus d'explication à divers contextes et en donnent des représentations courtes et claires (car déclaratives). Comme les opérateurs représentent actuellement la meilleure façon d'aborder la complexité de la génération d'explications, voyons comment ils peuvent être modélisés dans KADS.

1) Certains opérateurs sont indépendants du domaine de l'application et peuvent donc être modélisés comme des ***méthodes d'inférences*** (pour plus de détails sur ces dernières, cf [Breuker (ed) 87] [Weilinga 92]). Des recherches sont faites pour créer une bibliothèque de telles méthodes indépendantes du domaine [Safar 92].

2) Les autres opérateurs, et les concepts explicatifs qu'ils utilisent, doivent être modélisés dans le ***niveau domaine*** du ME-Communication. Dans ce cas, les ***méthodes d'inférences*** des SC ont pour but de ***contrôler*** l'application de un ou plusieurs opérateurs de domaine.

Dans les deux cas, les SC peuvent être plus ou moins complexes : les méthodes d'inférences d'une SC peuvent être **simples** et la SC **appelée plusieurs fois** successivement; c'est alors le contrôle au niveau tâche qui est plus compliqué. Nous laissons au concepteur ce choix de modélisation.

Nous allons nous appuyer sur ce principe de modélisation d'opérateur pour présenter, avec la figure 4 ci-dessous, une structure d'inférence possible ***pour la construction et le raffinement de la structure intentionnelle*** (centres des travaux actuels sur la génération d'explications). Il ne s'agit pas d'une solution idéale, mais d'une ***synthèse dans une formalisation KADS*** de travaux actuels **combinant** diverses méthodes de génération de discours : [Suthers 91], [Zukerman 91], [Lester 91], [Lemaire 91].

Toutes les étapes possibles dans la génération ont été retenues. Bien que toutes ces étapes soient **utiles** pour avoir de «bonnes» explications, la plupart des travaux actuels en omettent un certain nombre. Lorsque les quatre travaux de référence ci-dessus divergent sur l'ordre de certaines étapes, celui-ci n'est pas précisé. Ce sera alors **au concepteur de le définir dans le niveau tâche** en fonction de son application et de ses exigences en explications. Dans [Lemaire 91] les auteurs conseillent un opportunisme total, ce qu'ils réalisent en gérant les opérateurs via un blackboard. Le concepteur peut adopter cette solution de la souplesse optimale, mais du plus long temps de génération, en modélisant au niveau tâche une gestion de l'opportunisme.
Cette structure d'inférence est ***destinée à des explications complexes*** i.e. naturelles, adaptées à l'utilisateur et à ses buts, etc. Le concepteur peut donc éventuellement n'en prendre qu'une sous-partie *pour des explications plus simples.*

Enfin, comme il s'agit d'une synthèse, l'**utilité** des divers opérateurs à employer et surtout la **manière de les construire** ne sera **pas détaillée** ci-dessous. On se reportera pour cela aux auteurs cités. ***Seule leur mise en oeuvre dans le cadre de KADS et de l'architecture proposée sera discutée.***

Dans cette figure 4, pour des raisons de clarté, des appellations plus précises que «structure-de-données» ont été données aux rôles. Les noms des SC ont aussi été explicités.

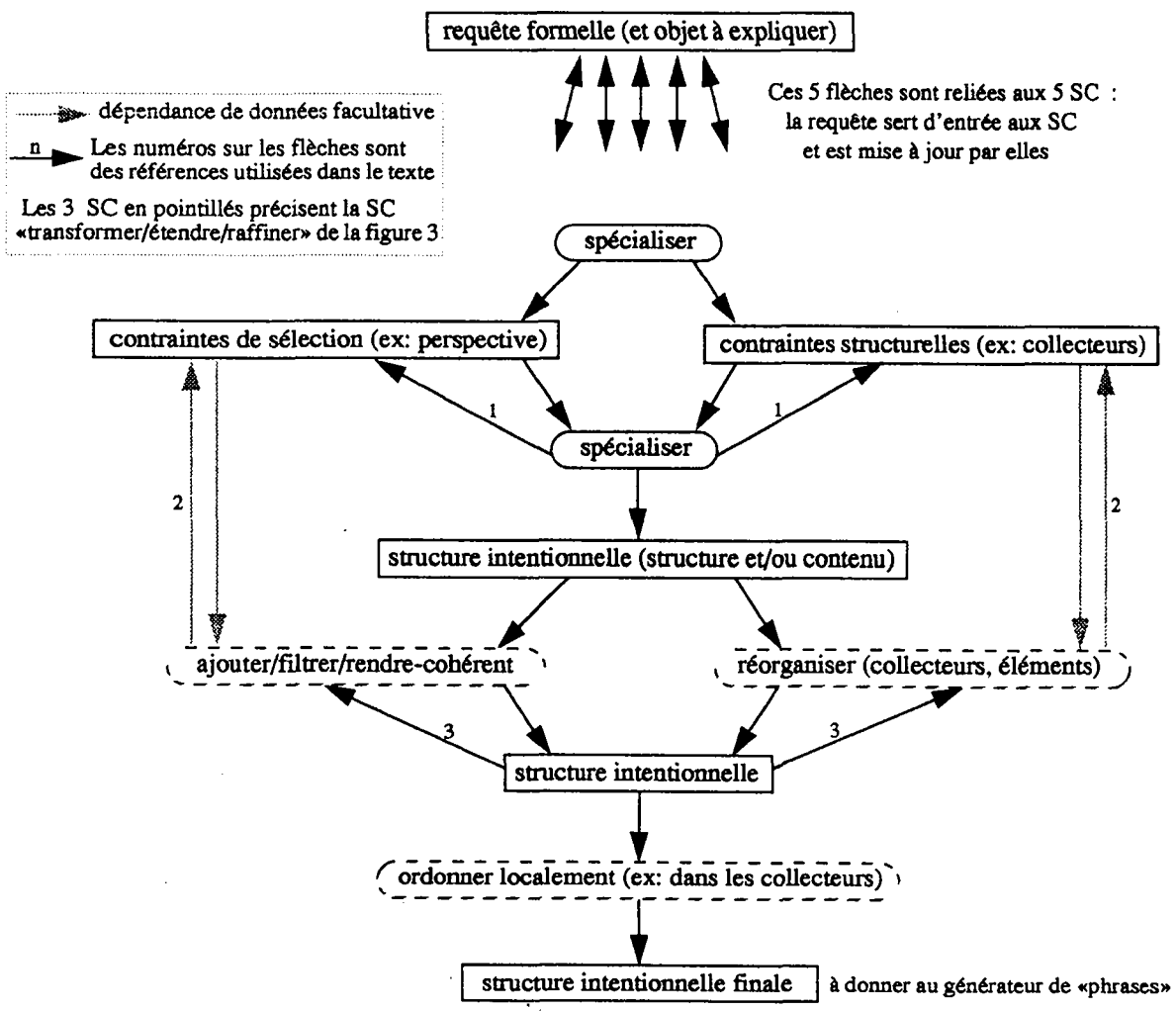


Figure 4: Précisions sur la construction de la structure intentionnelle

En génération d'explication, les opérateurs sont déclenchés pour atteindre un but ou sous-but de discours. Dans la structure d'inférence suivante, les *cinq* SC trouvent ces buts dans la requête formelle d'où partent donc *cinq flèches*. Ces flèches sont *bi-directionnelles* car chaque SC peut modifier la requête ou surtout la spécialiser pour la SC suivante. Dans la structure d'inférence précédente, cette notion de mise à jour de but n'apparaît pas explicitement car la requête formelle est incluse et propagée dans les «structure-de-données».

Les trois dernières SC en pointillés, ou plus exactement groupes de SC, remplacent le groupe de SC «transformer/étendre/raffiner» de la figure 3.
Les deux SC «spécialiser» remplacent «sélectionner/assembler» : elles sélectionnent des connaissances dans le niveau domaine du ME-Présentation et dans les «connaissances de support pour les explications», les utilisent et les adaptent en fonction du contexte : l'utilisateur, le contenu de la requête, etc. Le terme «spécialiser» est donc plus juste que «sélectionner». Ces SC, ou inférences, fournissent ainsi des *contraintes structurelles et informatives et des éléments de réponses.* L'*assemblage* de ces derniers est implicite compte-tenu des contraintes structurelles.

- ***La première SC «spécialiser» symbolise la phase de recherche de contraintes structurelles ou informatives***, souvent liées aux données et préalable à la phase de planification. Plus précise que «contraintes structurelles» est l'idée de «*collecteurs*» («clusters» chez [Lester 91]). Chaque collecteur rassemble des éléments d'explication nécessaires pour atteindre un but de l'explication. La première inférence peut donc fournir en tant que *plan initial* de l'explication une liste de collecteurs contenant des éléments. Exemple de plan initial : «Motivation», «Introduction», «Méchanismes». Plans et éléments peuvent être ordonnés mais ces derniers le sont rarement à l'issue de cette première phase.
  Cette prédétermination d'éléments et de contraintes mériterait d'être plus détaillée, mais elle dépend trop du type de la requête et de l'application pour cela. On peut se reporter à [Suthers 91] ou [Lester 91] pour plus de détails sur la détermination de la «*perspective*».

- ***La seconde SC «spécialiser» représente la phase d'expansion de buts (planification).*** Les opérateurs de planification sélectionnent, adaptent et assemblent des éléments de réponse, en tenant compte de *contraintes structurelles et de sélection.* Les **flèches «1»** sur la figure 4 montrent que les contraintes peuvent être mises à jour après chaque application d'opérateur. Ces flèches forment deux boucles, ce qui rappelle que cette seconde SC peut soit modéliser toute la planification, soit une inférence beaucoup plus simple mais appelée plusieurs fois successivement par le niveau tâche.
  Les **flèches «2»** grisées montre que le niveau tâche peut revenir à une phase de planification après une phase de correction. La série d'appels aux opérateurs de planification et leur alternance avec les opérateurs de correction sont donc gérés au niveau tâche, selon un mécanisme et des règles que choisira le concepteur : opportunisme total guidé par des heuristiques ([Lemaire 91]), alternance liées au opérateurs, alternance stricte, etc.
  Les flèches «1» et «2» permettent d'utiliser la même structure d'inférence quels que soient les différents séquencements des phases de planification et de correction. Voici quelques justifications de la nécessité de ces séquencements :
  1) il doit être possible de lancer une *phase de correction au cours de la planification* pour corriger les erreurs le plut tôt possible (donc après chaque application importante d'un opérateur ou d'un groupe d'opérateur);
  2) de façon à être interruptible, la planification doit commencer à *«présenter» des explications alors que* ***tout l'arbre*** *de leur génération n'est* ***pas*** *encore* ***développé***;
  3) la phase de planification elle-même peut s'effectuer en deux passes, l'une pour déterminer le contenu des explications et l'autre pour structurer ce contenu (mais à notre connaissance, seul [Zukerman 91] opère ainsi).
  Il y a donc un très grand nombre de structures de tâches possibles, i.e. de moyens de gérer ce

contrôle : structures de contrôle classiques, blackboard, etc. Aucune ne peut être jugée meilleure que les autres; celle de l'***opportunisme*** total étant probablement la solution la plus complète mais la plus coûteuse en temps d'exécution. Le contrôle doit inclure la gestion des ***retours arrière*** : exploration d'une autre branche de génération d'explication lorsque celle en cours apparaît inadéquate (grâce à des ***heuristiques d'évaluation***). Cette ***exploration*** peut se faire en «largeur d'abord» (e.g. [Lemaire 91]) ou en «meilleur d'abord», ce qui est la solution la plus commune.

- ***Les opérateurs de correction améliorent le contenu et l'organisation des éléments d'explication qui constituent la structure intentionnelle.*** De nombreux exemples de ces opérateurs sont donnés dans les quatre travaux de référence ici synthétisés.
  La structure intentionnelle est associée à une *mémoire des buts poursuivis pour sa génération* (e.g. arbre de génération de [Moore 91]) et suit les *contraintes structurelles* définies plus haut. Par exemple, ces contraintes peuvent être des collecteurs, lesquels peuvent être triés, divisés, ajoutés, etc., tandis que leurs éléments peuvent changer de collecteur, être supprimés, etc.
  Dans la figure 4, ces diverses inférences ont été regroupées par type d'action, mais il y a en réalité autant de SC que d'applications d'opérateurs différents. Les deux boucles (**flèches «3»**) permettent de modéliser les divers séquencements des applications de ces divers types d'opérateurs. C'est au niveau tâche de gérer ces séquencements, éventuellement de manière totalement opportuniste.
  Comme le signalent les flèches grisées, ces applications se font sous des contraintes et modifient ces contraintes ***pour orienter*** l'application future d'autres opérateurs.

- ***Lorsque plus un opérateur ne peut s'appliquer sur (la partie à développer de) la structure, ou que le niveau tâche ne le permet plus, commence une phase d'ordonnancement local des éléments de cette structure intentionnelle.*** Selon [Lester 91], ce peut être à trois niveaux : «entre intentions, entre propositions ou entre rôles d'une proposition». Les opérateurs d'ordre ou d'ordonnancement peuvent être selon [Suthers 91] des méthodes de : *traversée de graphes* (i.e. suivi de chaîne temporelle, causale, etc.), *d'exploitation de relations pédagogiques* (e.g. dans l'explication d'un concept, les définitions ou les généralisations sont en premier tandis que les illustrations ou les particularisations sont en dernier), *de dérivation de focus* (i.e. comment placer tel élément compte-tenu que tel autre est plus important), etc.

- La (***partie de***) structure intentionnelle ainsi développée peut alors être donnée au générateur de phrase (cf figure 3).

En résumé, pour modéliser l'expertise de présentation, le **niveau domaine** du ME-Communication doit non seulement contenir les différents concepts et relations relatifs aux langages (lexique, syntaxe, etc.) et aux «connaissances de support pour les explications» mais aussi les opérateurs qui exploitent ces connaissances. Les niveaux **inférence, tâche et stratégie** contrôlent les applications de ces opérateurs (cf chapitre 4 pour une synthèse des techniques actuelles de génération d'explication).
Nous n'avons pas mentionné dans les spécifications sur les divers contrôles possibles, ci-dessus, le niveau stratégie. En fait, pour reprendre sa définition, celui-ci doit adapter dynamiquement, i.e. en fonction du contexte ou des échecs, des structures de tâches - ces dernières étant les stratégies préfixées du niveau tâche. Cependant, dans le cas du ME-Communication, l'essentiel de ce trravail est déja effectué soit par le niveau tâche qui opère une gestion dynamique sur les opérateurs, soit par le ME-Coopération qui gère les problèmes de haut niveau de la communication : changement de l'objet de la transaction, des objectifs à atteindre, du type de dialogue, etc.

Il serait aisé de reprendre toutes les descriptions faites dans ce rapport sur les connaissances et méthodes que doit employer cette structure d'inférence (les deux inférences non redéveloppées incluses) ainsi que sur le contrôle qui doit la gérer, pour décrire un modèle d'interprétation (cf description des tâches génériques en annexe). Ceci n'a maintenant plus d'intérêt dans le cadre de ce rapport, mais les dites descriptions et la structure d'inférences peuvent être considérées comme **équivalentes à un Modèle d'Interprétation pour la Présentation**. Certes celui-ci est d'assez haut niveau et la modélisation de l'interprétation mériterait d'être bien approfondie, mais :
1) les requêtes réelles s'expriment souvent via un langage de requête peu ambiguë;
2) ce modèle peut déjà être une aide pour le concepteur lors de la modélisation de ses explications et l'acquisition des connaissances explicatives;
3) la plupart des «présentations» n'exigent pas des techniques aussi élaborées que celles que nous avons vues (destinées à des explications «naturelles» et efficaces).

Ce modèle pourrait donc éventuellement être rajouté à la bibiothèque KADS des modèles d'interprétation réels i.e. génériques mais destiné à des domaines d'activité précis (dans ce cas, la Présentation).
Pour être un Modèle d'Interprétation pour l'Explication, ce modèle devrait inclure une modélisation plus fine du dialogue et de la collaboration entre les tâches de Présentation et celles de Dialogue. Cependant, les spécifications données dans ce chapitre (et dans le précédent), peuvent déjà aider le concepteur.

Pour conclure ce chapitre, rappelons ses apports :
- spécifications sur l'organisation de la coopération entre le(s) modèle(s) de résolution et ceux de communication;
- spécifications sur les connaissances nécessaires aux modèles de communication et sur les méthodes qu'ils peuvent employer;
- spécifications sur divers contrôles possibles.

Chapitre 6 Une méthode d'acquisition des «Connaissances Utiles pour les Explications»

1 Connaissances Utiles pour les Explications

Nous avons vu que les «Connaissances Utiles pour les Explications» sont :
1) des connaissances opératoires sur l'expertise de résolution (connaissances du domaine, stratégies, etc.) qui doivent être explicites et structurées; le Modèle d'Expertise (de résolution de problèmes) ou ME-Résolution rassemble ces connaissances;
2) des «Connaissances Complémentaires» pour expliquer et justifier celles du ME-Résolution;
3) des connaissances sur les règles de l'explication (opérateurs, conventions, stratégies, etc.) dont un certain nombre dépend du domaine de l'application; ces connaissances sont à placer dans les modèles de communication (Dialogue, Présentation).

Il n'est pas forcément naturel de séparer les extractions de ces trois types de connaissances.

- Pour toute connaissance (concepts, ..., stratégie), il faut acquérir auprès de l'expert les **principes** qui justifient son existence (qui pourront être exploités aussi bien pour la **résolution** que pour l'**explication**) et toutes les connaissances nécessaires pour l'expliquer (textes, figures, schémas explicatifs i.e. manières de l'expliquer, etc.).
- La recherche de principes «**expliquant**» la connaissance peut donner lieu à des connaissances plus complètes ou plus générales, exploitables pour la **résolution**. Des techniques automatiques ont été développées pour cela :
  - apprentissage à partir de «cas» ou de solutions expliquées ([Charniak 85]),
  - détection des incohérences et des régularités sur une base de connaissances ([Ayel 90]),
  - généralisation de modèles (causaux ou autres) plus ou moins spécialisés (ex: heuristiques) et validation auprès de l'expert ([Cordier 91]).
- La prise en compte des **explications pose des contraintes** tant sur l'**acquisition** des connaissances (techniques d'extraction appropriées, modélisation adéquate) que sur la **conception** (choix de techniques, outils ou formalismes appropriés).

Les explications sont donc pour l'acquisition des connaissance une charge mais aussi un moyen de mieux modéliser (**complétude, consistance, abstraction, généricité**) l'expertise nécessaire à la résolution. Ceci conduira, entre autres, à un code plus sûr, plus générique et plus maintenable.

Ainsi, les explications sont bien à prendre en compte dès la phase d'extraction des connaissances. Dans le cadre de la méthodologie KADS, une grande part de cette extraction intervient lors de la construction du modèle d'expertise (acquisition et modélisation des connaissances destinées à l'application) donc après avoir décidé de la coopération entre le système et l'utilisateur (quelles tâches sont affectées à chacun et quelles sont les tâches de transfert entre eux).

Comme les divers types (ci-dessus) de connaissances sont liés, une extraction des connaissances pour le ME-Résolution devrait aussi fournir des connaissances explicatives, et inversement, pour une extraction dans un but explicatif (les trois points ci-dessus le montre). Ceci n'est qu'un constat et non un conseil pour mélanger les extractions des divers types de modèles (d'autant qu'il

y a surement des techniques d'élicitation mieux adaptées pour certains types de ces connaissances). Mais cette constatation laisser à penser que la recherche d'un bon séquencement des phases d'extraction (pour les diverses modélisations) est plutôt guidée par les connaissances déja acquises que par une méthode stricte.

C'est pourquoi, dans la méthode proposée dans la section suivante, le séquencement donné n'est qu'indicatif. Ce n'est surtout qu'une **liste de «Connaissances Utiles à l'Explication» que l'on peut acquérir au cours de l'extraction de l'expertise.**
Cette liste est présentée sous forme de questions, mais le cogniticien peut en fait utiliser toutes les techniques d'élicitation qu'il désire pour obtenir les connaissances désignées (interviews, observations directes, Magicien d'Oz, etc.). L'extraction peut même se faire en plusieurs fois, par raffinement progressifs.

Dans la section 2, nous observerons comment les structures d'inférences données au chapitre précédent peuvent aider à acquérir les connaissances nécessaires aux modèles de communication (i.e. aux tâches de transferts).

2 *Acquisition des connaissances liées au modèle d'expertise*

KADS conseille pour acquérir l'expertise et la modéliser dans un modèle d'expertise, de choisir des modèles d'interprétation dans sa bibliothèque (ceux qui semblent le mieux correspondre aux types de tâche utilisés par l'expert). Cette «reconnaissance» est possible, puisque une première élicitation a eu lieu de façon à distribuer les tâches entre le système et l'utilisateur.
L'extraction suit alors ces modèles d'interprétation et en retour les adapte et les combine (en vue de l'application). La méthode proposée s'inscrit dans ce cadre : même s'il n'est pas toujours possible de reconnaître des modèles «adéquats», la liste (sus-désignée) reste représentative de ce que l'on peut obtenir.

Les questions ci-dessous essaient de prévoir de nombreux cas. En contrepartie, certaines ne seront pas pertinentes dans un cas réel. Celles qui le sont pourront être posées par le cogniticien à l'expert, moyennant une **adaptation au domaine** de l'expertise.

Pour modéliser la connaissance de l'expert, le cogniticien utilisera le langage KADS : méta-classe, inférence, structure de tâches, concept, instance, relation entre concepts, relation entre propriétés de concept, etc. Or, pour éviter les biais, les questions posées à l'expert ne doivent inclure aucun des éléments du langage de la modélisation. S'il est relativement aisé d'éviter de parler de tâches ou de sources de connaissances, il est beaucoup plus dur de ne pas demander «des valeurs par défaut» ou «des valeurs qui seraient héritées de telle classe à telle sous-classe».

Pour aider le cogniticien à ne pas se laisser influencer et donc à mieux s'adapter au langage de l'expert, les questions ci-dessous sont rédigées dans des termes compréhensibles de tous (les conseils de construction du modèle, s'adressant eux, au cogniticien, sont dans les termes KADS). C'est pourquoi, les questions proposées sont assez générales. Il faudra donc de nombreuses questions de types comparables pour extraire des informations modélisables précisément. Lorsque des termes comme «concepts, rôles, classes ou catégories de concepts, méthodes, relations, etc.» sont inclus dans les questions ci-dessous, c'est bien sûr dans des sens beaucoup plus large que ceux donnés par KADS.

Voici donc cette proposition de méthode d'acquisition Rappelons préalablement que les ST sont des «Sources de connaissances Types», i.e. les SC «modèles» des modèles d'interprétation, et non les SC qui vont réellement être modélisées.

POUR tout modèle d'interprétation MI choisi pour remplir un modèle d'expertise ME
demander le but, l'intérêt et la justification de la ou des méthode(s) réellement utilisée(s)
essayer de remplir et d'adapter les ST et les méta-classes de MI (cf «A:», «B:» et «C:» ci-dessous; ainsi le niveau inférence de ME est mis à jour, son domaine rempli et les «connaissances explicatives qui lui sont associées» aussi)
questionner sur le contrôle et les stratégies d'utilisation des inférences exhibées à partir des ST de MI (les questions «A» peuvent être facilement adaptées pour cela; le niveau tâche de ME est ainsi mis à jour)
questionner sur l'utilisation de la tâche (contexte de déclenchement, etc.; ici aussi, les questions «A:» peuvent être utilisées; le niveau stratégie de ME est ainsi mis à jour)

A: remplir ou adapter une ST (qui rappelons-le, est une fonction élémentaire dont les entrées/-sorties sont des «méta-classe» ou «rôles») pour modéliser une SC

Compte-tenu du contexte de la ST (e.g. sa position dans la structure d'inférence de MI)
demander si une telle fonction (modélisée par la ST) existe réellement
si oui quels sont ses buts, principes, avantages,
quand est elle utile et quand d'autres fonctions peuvent-elles être appliquées ?
si non, adapter la ST ou ajouter une nouvelle ST à MI
remplir et adapter les méta-classes d'entrées et de sorties de ST (cf «B:» ci-dessous)
quelles sont les méthodes réellement employées pour réaliser cette fonction (ceci pour modéliser les méthodes d'inférence de la SC) ?
quelles sont les caractéristiques communes à ces méthodes ?
quels sont les critères de choix entre ces méthodes (e.g le contexte d'exécution) ?
pour chaque méthode, demander sa description et celle de son contexte de déclenchement :
- quelles sont les relations (du domaine : fonctionnelle, spatiale, etc.) qu'elle utilise et quelles autres relations pourraient éventuellement être utilisées (et quand) ?
- quelles sont ses étapes, ses sous-méthodes, ses liens avec d'autres méthodes, etc. ?
- quels sont les critères pour évaluer ses résultats, ses échecs ?
- en cas d'échec, quand l'interrompre et quelles actions de réparation entreprendre ?
- quelles sont ses limites, peut-on l'améliorer et dans quel contexte ?
- quels sont les autres buts qu'elle peut servir ?
- est-elle utilisée ailleurs (e.g. autre SC), autrement, peut-on généraliser son utilisation ?

Avec toutes ces informations, instancier la ST et compléter les «connaissances explicatives complémentaires associées à» ME; d'autres niveaux de ME peuvent également devoir être mis à jour.

B: remplir ou adapter une méta-classe

Compte-tenu du contexte de la méta-classe (e.g. sa position dans la structure d'inférence de MI)
demander si des concepts ou objets jouent un rôle similaire à celui de la méta-classe
si oui, demander leur caractéristiques communes,
si non, adapter la méta-classe ou ajouter une nouvelle méta-classe à MI
demander les caractéristiques communes à ces concepts ou objets
demander des précisions sur le rôle joué,
pour chaque concept ou objet, demander sa description et son contexte de déclenchement (on retrouvera bien sûr des concepts déjà décrits) :

- avec quels concepts ou classe de concepts est-il en relation (exemples de relations: composition, appartenance, causalité, définition, analogie, différence) ?
- quels autres rôles joue-t-il au cours du raisonnement ou d'autres raisonnements ?
- comment est-il affecté par ce(s) raisonnement(s) ?
- quelles sont ses propriétés, comment les conserver ou les faire évoluer ?
- quelles sont ses valeurs (par défaut) et les types de valeurs qu'il peut admettre ?

Avec toutes ces informations, instancier la méta-classe et compléter les «connaissances explicatives associées à» ME.

«C:», la troisième partie de cette méthode est plus destinée vers l'acquisition de connaissances destinée aux explications et est donc complémentaires à «A:» et «B». Elle peut se faire après ou pendant «A» ou «B».

C: extraire des méta-connaissances sur une information (obtenue par «A:», «B:» ou «C:»)

Pour toute information donnée par l'expert, e.g. concept, méthode, relation, critère, but, tâche, stratégie, etc. :
- de quels types d'utilisateurs est-elle famillière et l'utilisent-ils ?
- quelle est son importance dans le domaine ou dans la résolution (centrale, annexe, etc.) ?
- quels sont les multiples points de vues sous lesquels on peut la voir ?
- pour quels types d'explication, cette information est elle utile, dans quel contexte ?
- quelles sont les connaissances utiles pour l'expliquer : concepts (de résolution ou pas), relations, analogies, définitions (ex: synonymes), illustrations, principes qui la sous-tendent, etc. ?
- comment choisir entre les diverses connaissances possibles pour l'expliquer ?
- y a-t-il des façons classiques (schémas explicatifs) de l'expliquer ?
- comment choisir entre différentes stratégies pour l'expliquer : par référence à des justifications ou à la tâche mise en oeuvre, en reconstruisant un raisonnement, etc. ?
- si l'information est un choix : est-il bon, passable ou mauvais, a-t-il des alternatives, etc. ?
- et pour toutes les questions ci-dessus : 1) en fonction de quels types d'utilisateur ?
  2) dans quelle position par rapport à d'autres éléments de l'explication ou d'une explication plus globale (e.g. dialogue, etc.) ?
  3) dans le cas d'une explication au cours du raisonnement à quelle étape de ce raisonnement est-elle préférable (la réponse) ?

Avant de préciser (page suivante) comment l'exploitation de la sémantique des méta-classes et des sources de connaissances permet d'être encore plus précis dans la recherche d'informations, faisons quelques remarques.

- De nombreux travaux ont été effectués pour déterminer une **typologie des questions** que peut poser un utilisateur : [Dominguez 92] (pour associer une stratégie d'explication à chaque type de question), [Karsenty 92] (pour effectuer des explications spontanées), [Hugues 92] (pour mieux comprendre les requêtes), etc. Cependant, les questions réelles (et même souvent les questions types) ont généralement plusieurs rôles (ou buts) et appartiennent donc à plusieurs catégories. C'est pourquoi, [Gilbert 87] juge plus simple et plus efficace, pour la recherche des connaissances explicatives (faits, méthodes, stratégies, etc.), de dresser une **typologie des réponses** que l'utilisateur attend du système (et il montre la correspondance entre sa typologie de réponses et les diverses catégories des typologies de questions des travaux précédents).
- La **démarche proposée** ci-dessus s'inspire peu de ces typologies (assez formelles), mais **couvre** les diverses connaissances explicatives qu'elles impliquent. Son adaptation à la méthode KADS la rend plus utile que les typologies et elle est beaucoup **plus précise** (dans la recherche des connaissances).

- Entre autres conséquences, soulignons que les informations recueillies par cette démarche permettent de donner plusieurs vues sur chaque élément du ME-Résolution : *catégorielle* (comment un concept rentre dans une hiérarchie), *fonctionnelle* (rôle dans un processus), *structurelle* (parent ou fils structurels), *modulatoire* (comment un objet ou processus affecte ou est affecté par un objet ou processus), etc. Nous avons vu au chapitre 4 que ceci était utile pour la détermination de la «perspective» (sélection des concepts et des relations sur lesquelles se basera l'explication).

Afin d'aider le cogniticien à bien interpréter et abstraire le **rôle** des données lors de la résolution, voici des exemples de caractéristiques supplémentaires (associées aux groupes de méta-classes KADS) qui peuvent être sources de questions :
- problème : sa difficulté;
- intention : sa possibilité de réalisation;
- donnée : signification, taille, organisation, régularités, exceptions;
- rôle intermédiaire de donnée de problème : durée de la transition, place dans le raisonnement;
- rôle intermédiaire de donnée du domaine : support du raisonnement, standardisation;
- solution : unicité, distance par rapport à une solution idéale.

Dans le même esprit, mais associées aux **ST**, voici des types de questions qui pourraient ne pas être facilement déduits des questions génériques proposées.

Type de questions par ST

Considérons tout d'abord le groupe d'**inférences qui génèrent ou retouvent un concept** (ou une instance dans le cas d'«instancier»).

- **instancier** (concept -> instance), ex: trouver les valeurs d'un ensemble de paramètres, acquérir les caractéristiques d'un chien; pour les questions, nous nommerons entité cette instance utilisée :
  - comment acquiert-on (déduit-t-on) les valeurs ou caractéristiques de l'entité ?
  - que fait-on si certaines de ces valeurs ou caractéristiques ne peuvent être acquises ?
  - que faire si l'entité a des caractéristiques hors norme ?

- **identifier** (instance -> concept): c'est l'inverse de instancier, ex: «Fido est un chien», «tel phénomène ou appareil particulier peut être modélisé par tel concept»;
  - comment recherche-t-on et sélectionne-t-on le concept ?
  - que faire si l'entité ne peut être aisément classée (caractéristiques hors norme) ?
  - que faire en présence de deux concepts pertinents ?
  - comment comparer, pour cette classification, la pertinence de plusieurs concepts concurrents ?

- **généraliser** (ensemble d'instances -> concept): trouver un concept couvrant les caractéristiques des diverses entités présentées (si le concept doit déjà être connu, «généraliser» ressemble à «identifier»), ex: sur un historique de valeurs, remarquer qu'elles sont toutes impaires ou «proches» de telle valeur, etc.; pour l'expert, nous préfererons «objet» à «instance»
  - le concept est-il déjà connu, toujours connu ?
  - quels sont les critères de cette généralisation ?
  - quels sont les critères pour gérer les exceptions dans les valeurs ou caratéristiques des objets ?
  - que faire si des valeurs ou caractéristiques des objets ne peuvent être obtenues ?
  - que faire si plus d'un concept est un bon candidat pour la généralisation ?

- **abstraire** (concept -> concept): le 2ème concept doit avoir moins de propriétés que le premier (abstraction de certaines propriétés selon un certain point de vue), ex: un chien est un mammifère, un animal, un compagnon; un patient a de la fièvre si sa température dépasse 36 degrés celsius (abstraction qualitative); en pratique, il n'y a pas toujours abstraction, mais seulement changement de point de vue (l'identité du concept est inchangé);
  - quel est le type d'abstraction considéré ?
  - le 2ème concept est-il déjà connu, toujours connu ?
  - y a-t-il une transformation préalable du/des concept(s) avant son/leur abstraction ?
- **spécialiser** (concept -> concept): c'est l'inverse d'abstraire (et également appelé raffiner pour dénoter une longue chaîne d'inférences impliquant des spécifications); c'est de loin la ST la plus utilisée, avec sélectionner, sa forme dégradée (un concept spécifique est retenu); ex: modèle-du-système->spécialiser->norme, composants->spécialiser->spécifications-formelles, modèle-du-système->sélectionner->ensemble-de-paramètres, etc.
  - y a-t-il propagation de propriétés ou de spécifications ?
  - y a-t-il une transformation du premier concept ?

Pour les **inférences de comparaison**, **comparer** (valeur de X, valeur de Y -> différence) et **matcher** (structure X, structure Y -> concept-exprimant-la-différence-de-structure) :
- quels sont les divers types, concepts ou échelles de différences, comment classer le résultat ?
- certaines propriétés de X et Y sont-elles plus importantes que d'autres pour la comparaison ?
- y a-t-il des critères «abstraits» (de haut niveau) de comparaison ?
- y a-t-il une priorité entre ces critères, et si oui, quelle importance cela a pour le raisonnement ?

Pour les **inférences de modification de concept**,
assigner_valeur (attribut de concept -> attribut valué),
évaluer (structure X -> concept de X valué) :
- quel type de calcul est fait (qualitatif, numérique, etc.), quels sont ses principes, etc. ?

Pour les **inférences de modification de structure** (les plus génériques; le cogniticien remplacera «stucture» par l'entité considérée), rechercher les inférences plus primitives qui les sous-tendent :

- **assembler** (ensemble d'instances -> structure «partie-de»): réunit divers composants dans une seule structure (ex: plan, configuration, conception)
  - des éléments prédéterminés sont-ils combinés en satisfaisant des contraintes ou bien des éléments appropriés à certaines contraintes sont-ils choisis pour être insérés dans une structure prédéterminée ?
- **décomposer** (structure «partie-de» -> ensemble d'instances): c'est l'inverse d'assembler
  - quelles sont les contraintes, les éléments sont-ils prédéterminés ?
  - s'il y a plusieurs méthodes de décompositions, quelles sont les avantages de chacune ?
- **transformer** (structure X -> structure Y): trie ou restructure les éléments de X, ex: passer d'une structure linéaire à une structure hiérarchique (dans ce cas, l'entrée est une séquence, donc une structure, d'où la différence avec assembler), passer d'un formalisme à un autre, etc.
  - si la structure est modifiée, à quel degré ?
  - y a-t-il abstraction de la structure avant sa transformation ?

De la même manière que les modèles d'interprétation de la bibliothèque KADS peuvent guider l'acquisition des connaissances **opératoires (et possiblement explicatives** comme le montre la démarche ci-dessus), le Modèle d'Interprétation pour la Présentation (cf chapitre précédent) peut guider l'acquisition de connaissances **explicatives (et éventuellement opératoires)**.

La même démarche peut donc être employée. Dans la section suivante, des exemples de questions sont donnés.

3 *L'acquisition des connaissances explicatives*

Relevons d'après les figures du chapitre précédent, quels types de connaissances doivent être acquis (auprès de l'expert, de spécialistes en explication, etc.) pour le module d'explication.

- **Figure 1 et 2**, p 54 et 62 : les connaissances de *dialogue*, celles de *présentation*, et celles de *coopération* entre ces deux modules; les connaissances de *coopération* entre le ME-Résolution et le module d'explication ainsi que celles concernant la *gestion de l'ensemble des dialogues* (ou transferts avec l'utilisateur) e.g. application et adaptation du plan de session; comment *organiser, construire et mettre à jour* en interactif, les *connaissances de support* (même les Connaissances Complémentaires, peuventt être mises à jour en fonction de leur succès ou échecs e.g. en les *spécialisant* ou en leur *associant* les situations où elles s'avèrent utiles *pour tel ou tel type d'explication*).

- **Figure 3**, p 62 : quelles sont les *caractéristiques* du *langage* à utiliser pour les *requêtes*, son degré de *formalisation, son lexique, sa grammaire, sa sémantique, sa pragmatique* (conventions, etc.), etc. Les mêmes questions se posent pour le *langage formel* qui doit exprimer le contenu réel de la requête. Sous le terme langage, sont désignés tous les types d'expressions de requête : menus, associations de demande d'explications sur certains évènements, etc.
  Si la requête peut contenir des *éléments implicites ou inutiles*, comment les rechercher ?
  Comment s'effectue la *validation de l'interprétation de la requête*, dans quel *langage* et sous quel *contrôle* (la résolution ou la recherche d'explications est-elle lancée sans attendre la validation, et si oui peut-elle être interrompue; faut-il plusieurs modes de contrôle; etc.) ?
  Comment mettre à jour le *Modèle de l'utilisateur* et la *Mémoire de Dialogue* ?
  Comment relier les *éléments du ME-Résolution* avec ceux de la requête (ceci est sous-entendu dans la définition du langage formel, mais méritait d'être souligné).

- **Figure 4**, p 63 : quel *langage* utiliser pour répondre à la requête, quelle importance accorder à la *recherche d'informations* (résolutions intermédiaires, exploitation de la trace ou de la base de connaissances) *préalable au processus de génération d'explication* ?
  Comment en fonction de cette pré-recherche et de ce langage, *sélectionner et assembler* les éléments de réponses, *corriger cette structure intentionnelle*, et *générer des «phrases»* (texte, graphique, son, etc.) ? Ceci en s'*adaptant* au mieux à l'utilisateur pour servir ses *buts*.
  Comment mettre à jour les différentes *connaissances de support* ?

- **Figure 5**, p 65 : quelles sont les relations ou les primitives pouvant jouer le rôle de contraintes *structurelles* ou de contraintes *informatives* (des exemples ont été donnés), quelles connaissances *pédagogiques, linguistiques*, etc. peuvent jouer ce rôle et fournir le fondement des divers *opérateurs* (planification, correction, etc.), comment construire ces opérateurs (suffisamment abstraits, avec ou sans préconditions mais toujours en relation avec le «*contexte*», etc. ; ceci au

niveau *domaine*, qui doit donc aussi contenir des «*concepts*» et des «*relations*» spécifiques de l'explication), quel degré d'*opportunisme* devront avoir ces opérateurs et comment gérer cette opportunisme (i.e. le *séquencement* des appels de ces opérateurs ou le contrôle de leurs *déclenchements*; ceci au niveau *tâche* voire au niveau *stratégie*), comment gérer (aux mêmes niveaux) le développement en parallèle des *alternatives* ou les *retours arrières* ?

Ainsi, le **cogniticien** (et/ou le **concepteur** du SBC) peut être **guidé** par les diverses Figures (les 3 dernières sont des structures d'inférences) en se demandant quelles connaissances doivent être acquises pour pouvoir construire les différents éléments de ces Figures.

Pour clore ce chapitre, rappelons que l'acquisition des stratégies explicatives dépend en partie de la conception du produit final et a des répercussions sur cette conception. Il s'agit de deux problèmes différents, mais ils ont des réponses identiques :
- toutes les connaissances (explicatives) ne peuvent être finement modélisées (et donc acquises) en indépendance totale avec les répercussions de certains choix de conception (interfaces, etc.) mais le maximum doit l'être;
- certains outils permettent de tester le Modèle Conceptuel (et donc de «voir» ses divers comportements), il est alors possible de gérer une certaine incrémentalité dans la construction de ce modèle et notamment de prendre en compte des répercussions des «explications» sur la «résolution» et inversement (comme le souligne [Voss 90], un certain prototypage est souvent intéressant même au niveau du Modèle Conceptuel).

[Lemaire 91a] et [Safar 92] détaillent une démarche incrémentale pour l'acquisition de connaissances ou stratégies explicatives (auprès de d'expert). Ils proposent également de construire des stratégies explicatives génériques et de les instancier et développer dans le cadre de l'application réelle. Ces stratégies génériques permettraient d'aider l'expert à (re)trouver **ses** stratégies explicatives personnelles.

Conclusion

Ce rapport a présenté une exploitation de la méthodologie d'acquisition des connaissances KADS pour produire des systèmes à base de connaissances (SBC) capables de fournir de bonnes explications de leurs connaissances et raisonnements à leurs utilisateurs.

Pour cela, des spécifications sur les connaissances et l'architecture de ces SBC ont tout d'abord été données (structuration, abstraction, multi-représentation, connaissances profondes, etc.). Puis, la présentation des méthodologies d'acquisition des connaissances et en particulier KADS, a montré l'intérêt de ces spécifications pour toute conception de SBC, et l'importance de modélisations intermédiaires pour les prendre en compte lors de l'extraction et de l'organisation des connaissances.
Une description détaillée et homogène de KADS, de ses apports et de ses faiblesses a été donnée : chapitre 3, chapitre 5 et Annexe.

De même, les expertises liées aux explications a été introduite par une synthèse des différents types d'explication, de leurs rôles et des techniques ou stratégies explicatives actuellement employées pour ceux-ci.

Ces deux synthèses posent des spécifications sur la construction de SBC explicatifs, donc la guident et permettent de comprendre comment il est possible de raffiner pour l'explication la méthodologie KADS. Le chapitre 5 propose et spécifie des modèles d'expertise pour :
1) le contrôle de la coopération entre le système et l'utilisateur (ME-Contrôle);
2) la gestion des communications avec l'utilisateur : dialogue, présentation (interprétation d'une donnée ou d'une requête de l'utilisateur; présentation de données à lui faire connaître).

Le ME-Contrôle synchronise et contrôle le ME-Résolution et le ME-Communication. La réunion de ces trois modèles forme le Modèle Conceptuel (de l'expertise nécessaire à l'application).
Des «connaissances de support» ont été spécifiées pour ces différents modèles : quand et en quoi elles sont nécessaires pour l'exécution des connaissances de ces modèles.

Pour le ME-Communication, diverses structures d'inférences ont été proposées. Elles permettent d'obtenir, en prenant en compte les spécifications des chapitres précédents, un modèle d'interprétation pour la Présentation.

Puis, le chapitre 6 a proposé une méthode pour recueillir les différentes connaissances *explicatives comme opératoires* nécessaires à la construction de ces différents modèles et de leur connaissances de support. Il est montré comment le concepteur peut être guidé dans l'extraction des connaissances par les diverses spécifications recueillies et celles proposées (notamment les structures d'inférences). Ainsi, ces spécifications sont bien une aide à l'acquisition des connaissances au sens large : **extraction, modélisation, conception.**

Une voie de recherche dans le prolongement de ce travail serait de spécifier plus précisément les divers modèles proposés (structures d'inférences, etc.) et leurs connaissances de support.

ANNEXE Les tâches génériques de KADS

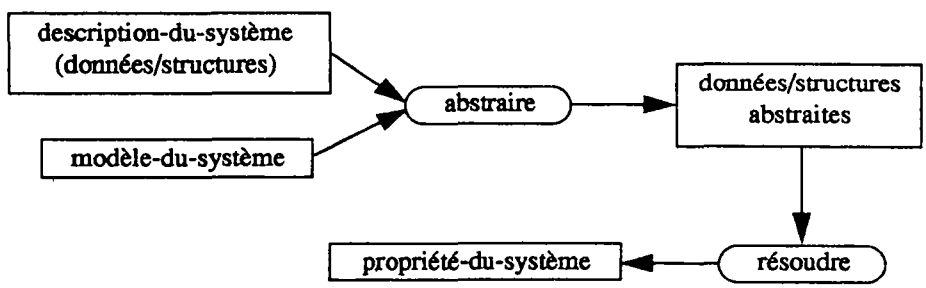
1 Les tâches d'analyse

Comme abordé au chapitre 3, pour ces tâches, la structure interne du système analysé est connue et la solution consiste en l'identification d'une propriété courante du système ou la «prédiction» d'un état de ce dernier (au sens d'identification d'un état actuellement non observable).

```
identification
      classification //identification d'un concept ou d'un cas
                                    comme un des concepts connus du modèle du système
          classification_simple //inférence identifier (ou généraliser ou abstraire)
            diagnostic
                  diagnostic_faute_unique
                        diagnostic-par-classification-heuristique //p 82
                        diagnostic_systématique //p 78
                              localisation structurelle //p 79
                              localisation causale //p 80
                  diagnostic_fautes_multiples
            évaluation (de correspondance, d'adéquation, etc.) //p 83
      surveillance //continue d'un système en fonctionnement //p 85

prédiction  //identification d'un état passé ou futur du système //p 89
      prédiction_de_comportement
      prédiction_de_valeurs
```

Chaque feuille de cette hiérarchie fait l'objet d'une tache générique (à part la classification simple qui n'est que l'appel à la ST *classifier*) et sera détaillé plus loin.

Dans les tâches d'analyse, les données (i.e. description de système réel en fonctionnement ou de cas réel) sont tout d'abord abstraites (interprétées) dans les termes d'un modèle du système étudié (ce modèle est pris dans la base de connaissances du SBC; il doit donc s'y trouver). Ce processus d'interprétation des données peut être plus complexe que ne l'évoque le schéma ci-dessous, lorsqu'il y a une forte intéraction entre l'abstraction des données et l'identification des structures (souvent implicites) de la description.



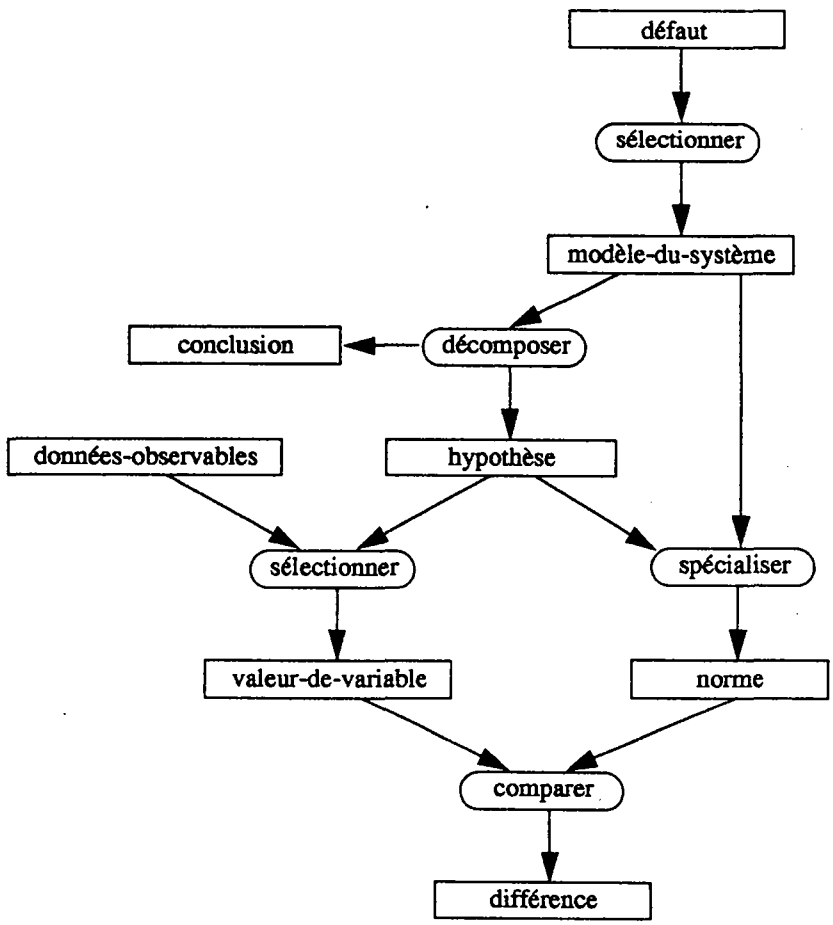
Le processus de résolution consiste en général à raffiner ces interprétations de façon à ce qu'un concept de classe (propriété du système comme par exemple. un défaut, un comportement, ... ou bien classe du système comme «acceptable» ou «inacceptable») puisse être trouvé comme solution du problème.

Les tâches les mieux connues sont celles de diagnostic. Comme le suggère l'intuition, elles consistent à trouver une faute dans un système, une faute étant la «négation» d'une fonction ou d'un composant correct. Comme l'univers des «négations» est bien plus grand que la description d'un système correct, la recherche des fautes est souvent basé sur l'expérience (priorité aux anomalies arrivant fréquemment). Ce type de connaissance est de nature heuristique et, comme le souligne [Clancey 84], le diagnostic par classification heuristique est le modèle le plus couramment trouvé dans les systèmes experts (et SBC).
Le diagnostic systématique, de nature plus algorithmique, peut être utilisé si le système peut être divisé en composants dont le fonctionnement est analysable, ou si un modèle causal des effets des processus internes au système est disponible. Dans ce cas, il est possible d'éliminer une hypothèse sans prendre en compte ses probabilités d'occurence.

La présentation des tâches générique suit celle de KADS (précisions, structure d'inférence, puis présentation des ST et des méta-classes, enfin stratégie et exemple de structure de tâche). Le document de référence est [Breuker 87], ici synthétisé (et homogénéisé, réorganisé, etc.), mais dont toutes les idées ont été reprises. L'évolution de ces tâches a été pris en compte ([Weilinga 92] a été pris comme dernière référence de KADS-I). Les modifications entreprises dans KADS-II ([KADS-II 92Lib], [Gobinet 92]) n'étant pas à l'heure actuelle stabilisées, ne sont pas reprises ici.

1.1 Diagnostic systématique

Voici la structure d'inférence commune à la localisation structurelle et à la localisation causale.



1.1.1 Localisation structurelle

Cette tâche est possible si l'on dispose d'un modèle du sytème structuré par des liens type «consiste-en» ou «partie-de» (e.g. dans les dessins, les spécifications ou lorsque les composants sont facilement distingables), si les composants sont testables indépendamment du reste du système, si l'on dispose des spécifications détaillées du comportement du système ou des valeurs de sortie. Un domaine typique de ces cas : les machines électroniques (ex: les ordinateurs).
Voici pour cette tâche le détail des inférences présentées dans la structure ci-dessus. La stratégie est présentée avec celle de la localisation causale.

ST **sélectionner** //un modèle du système; c'est la première inférence du diagnostic
(in *défaut* : un système défectueux ayant au moins un composant défectueux;
out *modèle-du-système* : ce modèle est structuré par des liens «partie-de» ou «consiste-en»;
une partie pertinente de cette hiérarchie peut être sélectionnée
)
méthodes : association directe;
connaissances du domaine impliquées : celles relatives aux comportement et
à la structure du système.

ST **décomposer** //le modèle du système en sous-modèles du prochain niveau de la hiérarchie
(in *modèle-du-système* : comme plus haut (ce ne sera plus répété dorénavant);
out *hypothèse* : (sous-)système contenant un composant défectueux,
conclusion : atteinte si le système ne peut plus être décomposé; une (des) partie(s)
du modèle responsable(s) du défaut ont été localisé(s)
)
méthodes : descente de la hiérarchie «consiste-en»;
connaissances du domaine impliquées : structure «consiste-en» du système.

ST **sélectionner** //une valeur de variable parmi toutes les sorties observables d'un composant
(in *hypothèse, données-observables*;
out *valeur-de-variable* : la valeur observée d'une des sorties d'un composant
)
méthodes : générer et tester;
connaissances du domaine impliquées : celles relatives aux méthodes de test.

ST **spécialiser** //le modèle du système pour,compte-tenu des valeurs ou fonctions en entrée d'un
composant, prévoir la valeur ou fonction en sortie de celui-ci (la norme)
(in *modèle-du-système, hypothèse*; out *norme*
)
méthodes : association directe;
connaissances du domaine impliquées : celles relatives au comportement du système.

ST **comparer** //la valeur attendue avec celle observée
(in *valeur-de-variable, norme*;
out *différence* : composant fautif trouvé ou pas,
utilisée par le contrôle pour itérer ou pas la procédure
)
méthodes : comparaison de valeurs ou de fonctions;
connaissances du domaine impliquées : signification de la différence (ex: intervals de
certitude, etc.).

1.1.2 Localisation Causale

Cette tâche est utilisable s'il est possible d'obtenir des données sur une part significatives des états impliqués dans le modèle et s'il est possible d'expliquer la fonction d'un système par des relations causales (ex: relations mécanique ou électrique dans un moteur; en médecine, les modèles causaux partiels, par exemple dans la circulation du sang, permettent en suivant cette tâche de distinguer entre des causes d'états pathologiques et des effets secondaires).
Dans la réparation de machine électronique, les deux types de localisations alternent souvent car un seul point de vue est généralement insuffisant pour éliminer des hyothèses.

```
ST sélectionner //un modèle du système; c'est la première inférence du diagnostic
   (in  défaut : un état anormal;
    out modèle-du-système : un réseau causal
   )
   méthodes : matching (heuristique ou pas) de l'état avec le modèle entier;
   connaissances du domaine impliquées : liens causaux entre états du modèle.
```

```
ST décomposer //le modèle du système en sous-réseaux
   (in  modèle-du-système;
    out hypothèse : sous-réseau où peut être trouvé l'état défectueux,
        conclusion : atteinte si le système ne peut plus être décomposé; un (des) etat(s)
                     responsable(s) ont été localisé(s): noeud(s) feuille(s) ou, pour certains
                      modèles compliqués, noeud(s) intermédiaire(s)
   )
   méthodes : association directe;
   connaissances du domaine impliquées : le réseau causal.
```

```
ST sélectionner //un état dont la valeur est observé
   (in  données-observables,
        hypothèse : en général, l'état de départ du (sous-)réseau;
    out valeur-de-variable
   )
   méthodes : générer et tester;
   connaissances du domaine impliquées : le réseau et celles relatives aux méthodes de test.
```

```
ST spécialiser //le modèle du système pour, compte-tenu des divers conditions, connaître l'état
                 de départ (la norme) d'un sous-réseau causal
   (in  modèle-du-système, hypothèse;   out norme
   )
   méthodes : association directe;
   connaissances du domaine impliquées : les normes du modèle du système.
```

```
ST comparer //l'état attendu avec celui observé
   (in  valeur-de-variable, norme;
    out différence : état fautif trouvé ou pas, utilisée par le contrôle
   )
   méthodes : comparaison d'état;
   connaissances du domaine impliquées : signification de la différence.
```

Stratégie

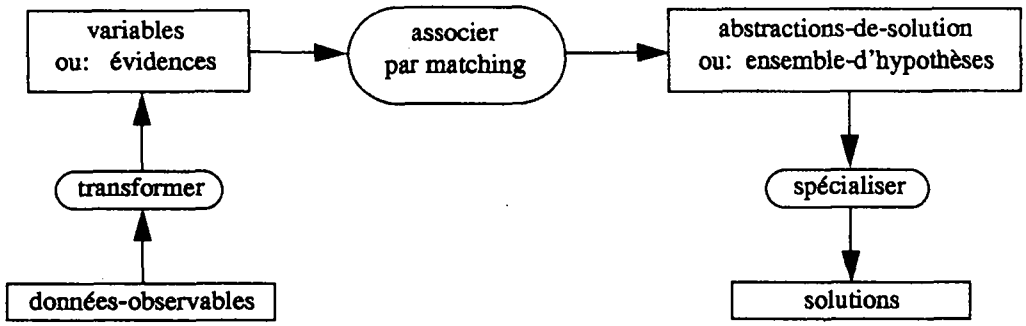
Voici, dans le formalisme présenté au chapitre 3, la structure de tâche du diagnostic systématique (les structures sont des stratégies fixées et ne représentent donc que des **exemples** de contrôle des tâches; il faut adapter et compléter ces exemples dans le cas d'une application réelle). Les ST sont en italique, les flèches séparent argument d'entrée et argument de sortie des sous-tâches, les tests et les arguments des tâches sont soit des méta-classes soit des termes de contrôles (ensemble de méta-classes).

```
tâche diagnostic-systématique
  but : trouver, s'il existe, le plus petit état incorrect (respectivement composant ayant un
                                                          comportement incorrect)
  termes de contrôle
     différentiel = ensemble des hypothèses en cours
     sous-système-incohérent = sous-partie du système ayant un comportement incohérent
  structure de tâche  diagnostiquer-systématiquement (défaut -> sous-système-incohérent) {
     sélectionner (défaut -> modèle-du-système)
     REPETER
         décomposer (modèle-du-système ->  différentiel,conclusion)
         POUR chaque hypothèse du différentiel
             spécialiser (modèle-du-système,hypothèse  -> norme)
             sélectionner (hypothèse,données-observables  -> valeur-de-variable)
               //une sous-tâche de transfert peut être utile pour obtenir la valeur
             comparer (norme,valeur-de-variable  -> différence)
             SI différence existe ALORS sortir de cette boucle POUR
    TANT QUE conclusion non atteinte //i.e jusqu'à plus de décomposition (affinement) possible
  }
```

Les diagnostics heuristique et systématique ne sont pas deux mondes séparés : les associations heuristiques peuvent d'abord être utilisées pour focaliser les causes d'anomalies les plus probables puis ensuite la localisation pour justifier, valider, apporter des preuves supplémentaires. Les modèles causaux peuvent en effet fournir des raccourcis dans le rasisonnement. D'autre part, les modèles causaux ou structurels (liens «partie-de», etc.) peuvent n'être que partiels et donc ne permettre d'observer que trop peu d'états, le raisonnement heuristique indique alors les probabilités des hypothèses restantes.

1.2 Diagnostic par classification heuristique

Ce type de tâche peut s'appliquer dans de très nombreux cas. Il est peut être trop général, car selon [Steels 90], la plupart des règles des systèmes experts pourraient être interprétés en fonction de ce modèle (plus exeactement, en fonction du modèle en «fer à cheval» de Clancey sur lequel est basé ce modèle d'interprétation). Voici sa structure d'inférences :



Voici, dans le domaine médical, des exemples pour les méta-classes de cette structure : *donnée-observable* (température de 40,2 degré celsius), *variable* (forte fièvre), *abstraction-de-solution* (infection bactérienne), *solution* (pneumonie due au pneumococcea).

```
ST transformer //les données observables en variables (abstraction des données); dans les cas
                simples, ce «transformer» est simplement un «abstraire»
   (in données-observables : si elles forment une structure, la méta-classe
            «description-de-choix» est plus précise et peut contenir «défauts» et «symptômes»;
    out variables : contenant les valeurs des paramètres du système en fonctionnement
   )
   méthodes : abstraction qualitative ou de définition ou par généralisation;
   connaissances du domaine impliquées : celles de définition.
```

```
ST associer-par-matching //(i.e. par comparaison de structure) les abstractions de données
                          avec d'autres concepts d'une autre classification; il s'agit de
                         corrélations typiques, heuristiques et parfois faiblement comprises
   (in variables;    out abstractions-de-solutions
   )
   méthodes : unification (i.e. comparaison de structure) heuristique;
   connaissances du domaine impliquées : règles heuristiques (type «cause-de», etc.).
```

```
ST spécialiser //ou raffiner, les abstractions de solutions en solutions plus spécifiques
   (in abstractions-de-solutions;  out solutions
   )
   méthodes : spécialisation (ex: descente dans une hiérarchie de lien «est-un»);
   connaissances du domaine impliquées : celles nécessaires à cette spécialisation.
```

Le diagnostic peut être dirigé par les données (depuis «obtenir-les-données» jusqu'au «spécialiser» ci-dessus) ou par les solutions (ordre inverse). Ce choix de contrôle est fait au niveau stratégie. Le cout de l'obtention des données est un facteur principal. S'il est élevé (en termes de temps, d'argent, de santé, etc.), une approche dirigée par les solutions est plus appropriée. Le choix du niveau de raffinement de la solution doit également être spécifié au niveau stratégie. Dans un cas d'urgence, un diagnostic sommaire est justifié.

1.3 Evaluation (de correspondance, d'adéquation, etc.)

Dans cette tâche, la description d'un cas est interprété en fonction des termes (ex: lois) d'un modèle du système afin de classer ce cas dans une des «classes de décisions» spécifiées à l'avance. Le classement s'effectue par rapport à des «normes» : combinaisons ET/OU de catégories ou de valeurs seuils pour des paramètres (les catégories étant des combinaisons de paramètres).

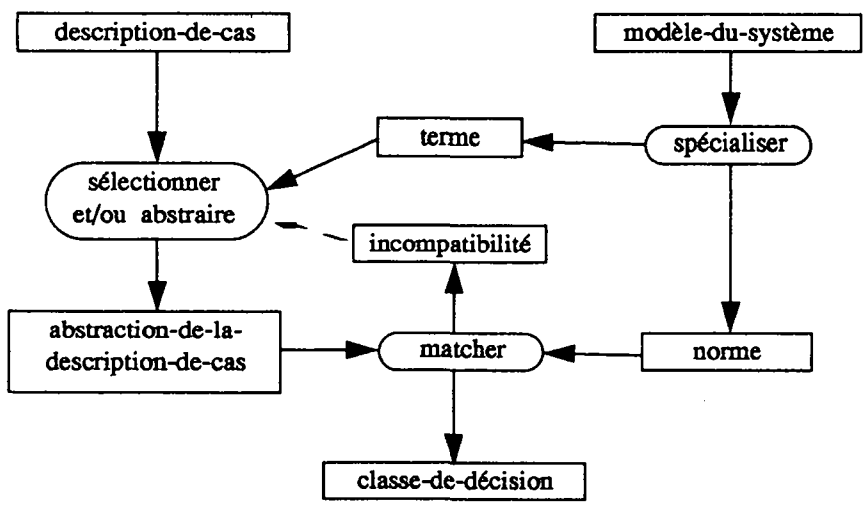
Voici quelques exemples : évaluation du type de lecteur pour la sélection des options d'un logiciel de bibliothèque, évaluation de la légalité du statut d'une companie par rapport à certaine lois, évaluation de l'adéquation d'un outil pour effectuer une tâche donnée ou bien d'une personne à occuper un poste.

Dans ce dernier cas, voici un exemple de ce que peuvent référer les méta-classes dans la structure d'inférence ci-dessous : *modèle-du-système* (description du poste), *terme* (expérience, connaissances, aptitude, ...), *norme* (au moins 3 ans d'expérience en T), *description-de-choix* (formulation de la candidature / lettre), *abstraction-de-la-description-de-choix* (expérience de X de Y années en Z), *incompatibilité* (T est un sous-ensemble de Z), *classe-de-décision* (adéquation ou non).

Ainsi, la tâche d'évaluation est plus complexe qu'une simple tâche de classification :
- l'interprétation du cas n'est pas donnée, elle doit être dérivée du modèle du système en concordance avec les nombreuses abstractions possibles des termes du cas traité;
- ceci implique des retours arrières et des alternances entre les deux lignes d'inférences (abstraction du cas, spécification du modèle du système);
- lorsque les cas ne sont pas standards, le niveau stratégie doit souvent recourir à un raisonnement plus profond, par exemple dans le diagnostic de l'écart entre le cas et le modèle du système : réinterprétation des fondements (principes) du modèle du système ou bien raisonnement par analogie (e.g. avec des cas précedemment traités, etc.

Notons que cette tâche générique est dans un problème réel, le plus souvent précédé ou alterné avec d'autres tâches (diagnostic, résolution, etc.). Il est très courant que plusieurs évaluations soient emboitées e.g. pour : vérifier la validité des données du cas, pour choisir entre plusieurs description de cas, etc.

Voici la structure d'inférence de cette tâche, commentée page suivante.



ST **sélectionner et/ou abstraire** //les attributs des données qui sont attributs des termes du modèle du système; s'il y a 2 ST, elles dépendent des mêmes connaissances : seuls les types prédéfinis sont reconnus; une approche plus flexible (où plusieurs points de vue sous-tendent le modèle du système) implique plusieurs inférences d'abstraction (ex: l'obtention d'un diplôme peut être vue comme l'achèvement des études ou le signe de certaines capacités ou connaissances); l'«incompatibilité» participe à cette mise en relation (cf structure de tâche);
outre l'abstraction des données individuelles, la structure qui les contient doit aussi parfois (e.g. dans les descriptions en langage naturel) être transformée (ST transformer)

(in *terme*,
description-de-cas : structure contenant des données individuelles (attributs valués);
out *abstractions-de-la-description-de-cas* : les (structures de) concepts sont remplacé(e)s par des concepts plus abstraits permettant l'application des normes (qui sont prévues pour des classes de cas)
)
méthodes : classification par héritage, association ou parsing (i.e. aux combinaisons de données sont attribuée des structures tout comme dans une analyse grammaticale);
connaissances du domaine impliquées : liens hiérarchiques et d'instanciation, règles de classification ou de «grammaire».

ST **spécialiser** //les concepts du modèle du système pour prendre en compte les cas individuels; plus les cas admis sont variables, plus il y aura d'inférences de spécialisation; s'il n'y en a pas, l'évaluation est simplement une classification

(in *modèle-du-système* : description des principes de classification, depuis la simple liste de classes jusqu'au modèle multi-niveaux (les plus hauts contenant les justifications); ex: esprit des lois, lois et procédures pénales;
out *terme* : catégories utilisées pour décrire les cas de façon à pouvoir utiliser les normes,
norme : (structures de) catégories (i.e. attributs avec leurs valeurs spécifiques)
)
méthodes : raffinement descendant, héritage (multiple) et, pour les domaines complexes (grande variété de cas, hiérarchies complexes), raisonnement causal ou temporel;
connaissances du domaine impliquées : hiérarchies de concepts.

ST **matcher** //(i.e comparer les structures) des abstractions de descriptions de cas avec celles des normes pour sélectionner une des classes de décision : acceptable, inacceptable, indécidable (échec partiel de la comparaison; l'évaluation doit alors se poursuivre éventuellement en raffinant pour lever l'ambiguité), etc.

(in *abstraction-de-la-description-de-cas*, *norme*;
out *classe-de-décision* : permettant de répondre à un concept (adéquation, responsabilité, ...) par des «oui/non/peut-être» ou des classements (des divers cas entre eux ou par rapport aux normes),
incompatibilité : écarts (quantitatif ou qualitatif) entre les structures; ils permettent d'obtenir de nouvelles spécialisations ou abstractions utile pour les réduire
)
méthodes : le choix est grand et dépend de la souplesse de la mesure; si l'on désire une stricte adéquation, une comparaison directe suffit; un raffinement descendant exhiber les différences les moins importantes en dernier et permet de traiter dans le même temps «abstraction» et «spécialisation» et donc de spécifier à quel niveau cas et normes matche ou pas
connaissances du domaine impliquées : critères de comparaison, règles, contraintes, restrictions de valeurs, associations de classes de décision à des normes.

Voici une structure de tâche typique. Elle reflète une approche descendante dans la mesure où les termes sont d'abord spécifiés, puis le cas interprété en fonction de ces termes. L'approche ascendante (interprétation puis comparaisons avec les termes du modèle) est plus souple mais a tendance à mener à des contradictions ou des divergences. Le contrôle entre ces deux approches relève du niveau statégie.

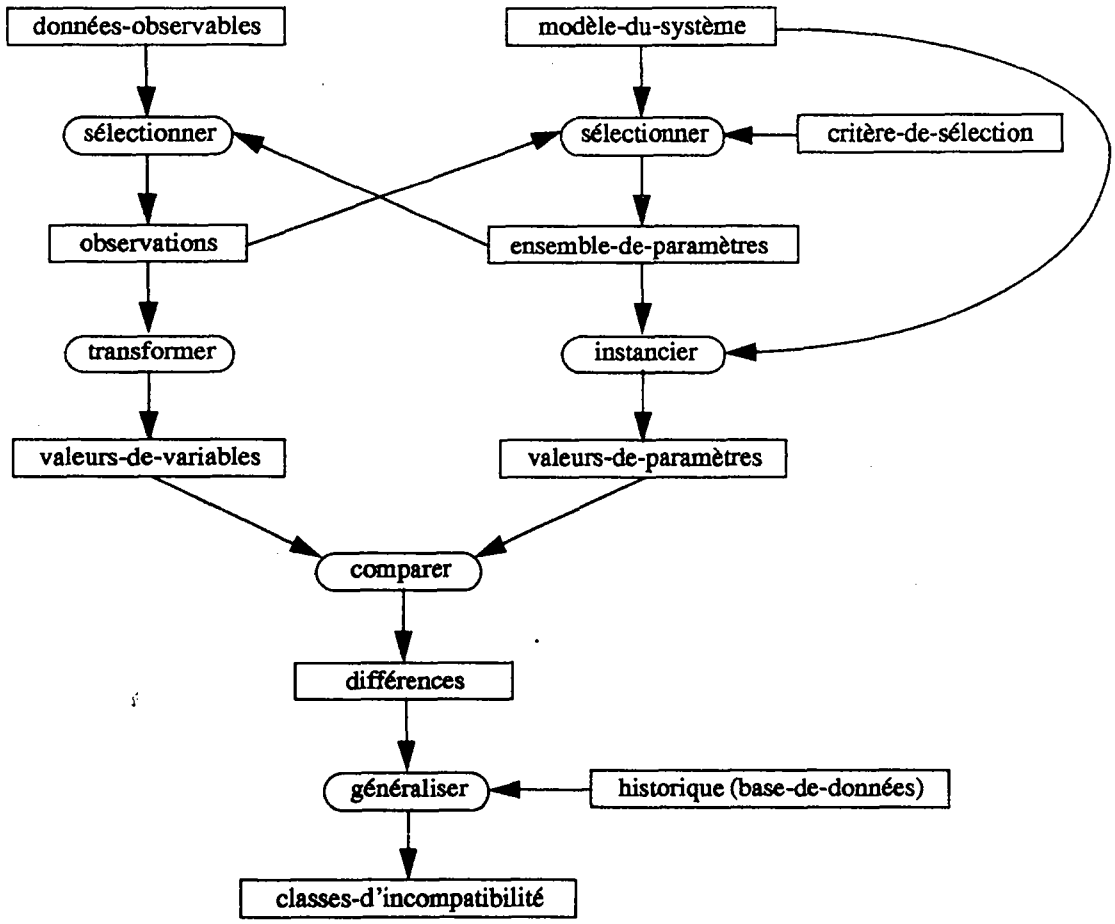
```
tâche évaluation
  but : déterminer si un cas respecte des normes prédéfinies
  structure de tâche évaluer (descripticn-de-cas, modèle-du-système -> classe-de-décision) {
     obtenir (description-de-cas)
     POUR tous les concepts de cette description-de-cas
         spécialiser (modèle-du-système -> terme,norme)
       filtrer-et/ou-abstraire (description-de-cas,termes -> abstraction-de-la-description-de-cas)
         //si aucun terme n'a pu être trouvé pour un concept, réagir en conséquence ici
     matcher (abstraction-de-la-description-de-cas,normes -> classe-de-décision,incompatibilité)
     //on peut matcher plusieurs fois pour affiner l'incompatibilité (focalisation)

     SI incompatibilité > acceptable ALORS
       //si un concept peut avoir plusieurs autres interprétations (termes), alors pour chacune
         spécialiser (modèle-du-système -> termes de l'incompatibilité, norme)
       filtrer-et/ou-abstraire (description-de-cas,termes -> abstraction-de-la-description-de-cas)
           matcher de nouveau
           ou obtenir de nouvelles données du cas
}
```

Les descriptions de cas réel étant rarement «typiques», nous avons vu que le niveau stratégie avait un grand rôle dans le contrôle des inférences et celui de leur alternance avec les inférences d'autres tâches. Sa tâche la plus fréquente est de réinterpréter les principes du modèle du système pour pouvoir traiter la description de cas présentée.

1.4 Surveillance (continue d'un système en fonctionnement)

Dans cette tâche, les données sont testées pour savoir si elles satisfont les normes. Si ce n'est pas le cas, une incompatibilité a été trouvée et suivant son type, telles ou telles décisions sont prises.

Ce modèle a été utilisé pour des applications allant du conrôle de processus au management de projet logiciel. La description des inférences qui suit est donc très générale. Pour l'étayer, des exemples sont donnés sur les applications suivantes : management d'un projet logiciel (PL) et surveillance de patients dans une unité de soins intensifs (SI).



ST **sélectionner** //un ensemble de paramètres dans le modèle du système
(in *modèle-du-système* : résultat d'une tâche de conception e.g. plan des activités d'un projet (PL), modèle patho-physiologique (SI), etc.,
critère-de-sélection : cf niveau stratégique (c'est de là qu'il vient),
observations : e.g. données historiques ou courantes sur les lignes de code produites par une équipe (PL), électrocardiogramme enregistré (SI);
out *ensemble-de-paramètres* : e.g. lignes de code source produites et nombre de bugs (PL), plan ou agencement d'électrocardiogramme (SI);
)
méthodes : générer et tester, arbres de décisions, squelette de plans, etc.;
connaissances de support : celles relatives au système.

ST **sélectionner** //un ensemble d'observations sur le monde réel
(in *données-observables, ensemble-de-paramètres* : ces derniers peuvent avoir une influence;
out *observations*;
)
méthodes : comme ci-dessus, tous les types sont possibles;
connaissances de support : celles relatives au monde réel.

ST **transformer** //les données observables du système en variables (abstraction des données);
(in *observations;*
out *valeurs-de-variables* : valeurs des paramètres du système en fonctionnement e.g.
fonction sur les lignes de source code actuellement produites (PL),
ou charactéristiques de l'électrocardiogramme (SI)
)
méthodes : abstraction qualitative ou de définition ou par généralisation;
connaissances de support : celles de définition.

ST **instancier** //l'ensemble de paramètres i.e. déduire leurs valeurs attendues
(in *ensemble-de-paramètres,modèle-du-système;*
out *valeurs-de-paramètres* : e.g. lignes de code source qui auraient du avoir été
produites à l'heure actuelle (PL) ou les charactéristiques
désirées de l'électrocardiogramme (SI)
)
méthodes : spécialisation de squelette de plan, etc.;
connaissances de support : celles relatives au système.

ST **comparer** //les valeurs de paramètres désirés avec celles observées
(in *valeurs-de-paramètres, valeur-de-variable;*
out *différences* : e.g. nombre de lignes de code source qui n'ont pu être produites (PL) ou
charactéristique anormale de l'électrocardiogramme (SI)
)
méthodes : comparaison d'attributs, soustraction s'ils sont numériques, du type comparaison
de chaînes sinon;
connaissances de support : ?.

ST **généraliser** //ou classifier (synonyme KADS de la méta-classe «généraliser») les différences
en classes d'incompatibilité à la lumière d'informations déjà stoquées dans
une base de données (comme que par exemple, le résumé des cycles précédents)
(in *différences, historique;*
out *classes-d'incompatibilité* : valeur ou sémantique de l'écart
)
méthodes : association heuristique;
connaissances de support : historique stoqué dans une base de données.

Il y a deux types de façons de lier ces inférences :
- soit l'approche dirigée par le modèle où le système a l'initiative et la tâche généralement effectuée à intervalles de temps réguliers (acquisition de données pour un ensemble choisi de paramètres et comparaison des valeurs obtenues avec celles attendues)
- soit l'approche dirigée par les données où la tâche est déclenchée par la venue de données : elle contient une tâche de transfert «recevoir» donnant l'initiative à un agent extérieur (humain ou autre système); le traitement de ces données est ensuite identique.

D'où les deux structures de base (où les «...» sont les paramètres d'entrées des inférences) :

```
tâche surveillance-dirigée-par-le-modèle
  but : exécuter un cycle de suveillance dans lequel le système acquiert les données
  termes de contrôle
     param-actifs = ensemble de paramètres
  structure de tâche  surveiller (modèle-du-système -> sous-système-incohérent) {
     sélectionner (... -> param-actifs)
     POUR chaque paramètre de param-actif
        sélectionner (... -> observations) / obtenir (données-observables)
                                          | instancier (... -> valeurs-de-paramètres)
                                          | transformer (... -> valeurs-de-variables)
              //le «|» indique que ces (quatres) sous-tâches peuvent se faire en parallèle
        comparer (... -> différences)
        généraliser (... -> classes-d'incompatibilité) //ou en dehors de la boucle pour un
     fin-pour                                          //traitement plus global
  }
```

```
tâche surveillance-dirigée-par-les-données
  but : exécuter un cycle de suveillance quand de nouvelles données sont reçues par le système
  structure de tâche  surveiller (modèle-du-système -> sous-système-incohérent) {
     sélectionner (... -> observations)
     sélectionner (... -> ensemble-de-paramètres) | recevoir (données-observables)
     POUR chaque donnée de données-observables
        transformer (... -> valeurs-de-variables) | instancier (... -> valeurs-de-paramètres)
        comparer (... -> différences)
        généraliser (... -> classes-d'incompatibilité) //ou en dehors de la boucle pour un
     fin-pour                                          //traitement plus global
  }
```

De manière plus générale, le niveau stratégie contrôle l'exécution de la structure de tâche en fonction du modèle du système, des classes d'incompatibilité. Il détermine ainsi

- le choix de la structure de tâche; exemple de règle : SI l'obtention des donnée est simple et coûte peu ALORS choisir l'approche dirigée par les données;
- le critère de sélection i.e. des règles permettant de choisir quels paramètres doivent être sélectionnés pour la surveillance (pour tous les cycles ou pour chacun d'eux);
- le temps de cycle, crucial pour une tâche de surveillance car elle doit s'effectuer à intervalles réguliers dans le temps e.g. toutes les semaines (PL), toutes les secondes (SI), etc.
  Le niveau stratégique peut modifier cette durée, par exemple la raccourcir si des incompatibilités sont détectées;
- suivant les classes d'incompatibilité, la décision de lancer une tâche de diagnostic et/ou de réparation.

1.5 Prédiction (ou détermination de ce qui va arriver dans telle situation)

Pour cette tâche, deux sous-types ont été identifiés : la prédiction de valeurs et celle de comportements. Le premier est la dérivation ou le calcul de valeur d'attributs à partir de valeurs d'autres attributs en relation (ex: résolution de problèmes physique ou autres tâches d'éducation). Le modèle sous-jacent du système est souvent formel et quantitatif.
Le second correspond aux tâches qui répondent à des questions comme «qu'est ce qui arrivera(it) si ... et quand ?». Le modèle est alors souvent qualitatif. Ce type de tâche est d'ailleurs souvent utilisé en raisonnement qualitatif.
Les deux tâches génériques correspondantes sont très détaillées dans [Breuker 87] (beaucoup plus que toutes les autres). Par homogénéité avec la présentation des autres tâches et parce que leur apport dans ce rapport serait assez minime, nous ne présenterons pas ces tâches, mais simplement celle, plus générale, communes à toute les tâches de prédiction.

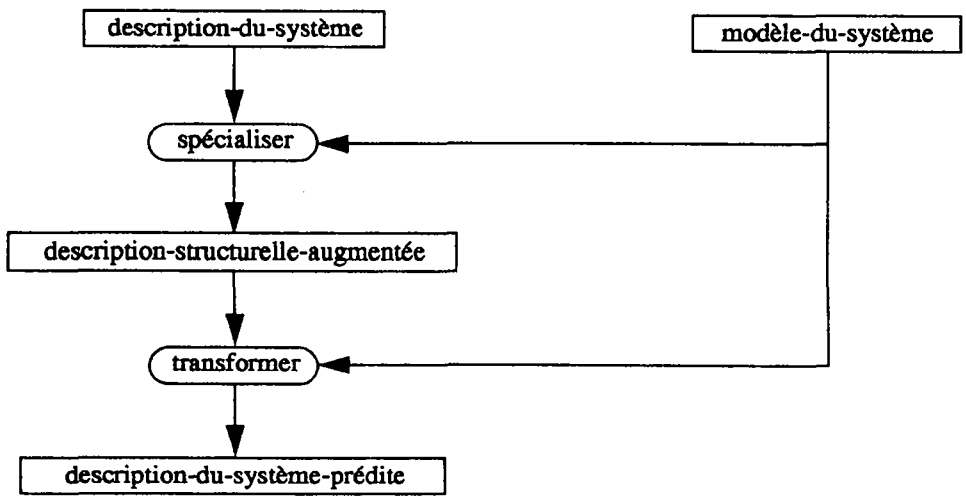
Ainsi, étant donné la description d'un sytème (actions ou processus et objets impliqués dans une situation donnée), la prédiction est soit la description de l'état final, résultat des intéractions entre processus et objets du système, soit une séquence de descriptions analogues, débutant par la description de l'état initial.

Une description peut être formelle (ex: en logique du premier ordre) ou pas (ex: savoir de sens commun sur le monde). Elle décrit une situation durant une période de temps (t1->t2) durant laquelle le comportement ne change pas (il se poursuit, ex: «l'eau gèle»; les objets peuvent se modifiés). Pour décrire de plus grande périodes de temps, il faut utiliser des séquences de description. Voici un exemple de description de système dans le monde (de la) physique :

Quelque part sur une colline, il y a une petite maison avec un toit en pente. Elle est prolongée dans le sens de la pente par une serre. Pour une raison quelconque, une boule de métal roule le long du toit de cette maison.

Pour effectuer certaines inférences prédictives, il faut savoir que dans de telles conditions la boule roule vers le bas, qu'elle peut ainsi atteindre la serre, que celle-ci comme la plupart des serres est probablement en verre, que la boule peut donc la traverser (si elle est assez lourde) et l'empêcher d'atteindre le bas de la pente. Si la serre n'est pas sur le chemin de la boule, peut-être une gouttière l'interceptera-t-elle, etc. Des connaissances venant du modèle du système sont donc utilisées.

Voici une illustration de la structure d'inférence d'une tâche générique de prédiction :



Les trois méta-classes de description du système ont été distinguées par des noms différents mais réfèrent au même type d'information : description de processus, d'objets et de leurs intéractions. Quant aux ST, elles ne sont pas auusi primitives que leurs homonymes présentées dans la typologie du chapitre 3. Cependant leurs noms sont assez intuitifs et les types d'inférences qu'elles représentent ressemblent grossièrement à ceux de la typologie.
Les exemples donnés dans les deux ST ci-dessous ne représentent pas l'interprétation d'un cas particulier mais illustrent juste le processus du raisonnement.

```
ST spécialiser //le modèle du système pour augmenter la description du système, en attributs
                comme en valeurs ou relations entre concepts
   (in description-du-système : cf exemple du monde physique,
       modèle-du-système : connaissances sur les processus, les objets et leurs interactions,
                           dans notre exemple, savoir général sur les serres, la résistance du
                           verre, les propriétés d'une boule de métal, de la pesanteur, etc.
    out description-structurelle-augmentée : dans notre exemple, des connaissances inférées
                  comme la serre est en verre, la balle suit la direction de la pente et va donc
                  atteindre la serre, donc en supposant qu'elle tombe de 1m50 et qu'elle pèse ...
   )
   méthodes : raffinement, héritage, association heuristique, généralisation, etc.;
   connaissances du domaine impliquées : hiérarchies de concepts, relations causales ou de
              déclenchement, règles de classification, ... suivant le domaine (très variable).

ST transformer //la description-structurelle-augmentée en une description prédite (état du
                système suivant l'état courant)
   (in description-structurelle-augmentée, modèle-du-système;
    out description-du-système-prédite : tout ou partie de l'état final, car ce ne peut être une
                           séquence; mais elle peut servir à une nouvelle tâche de prédiction;
                            par exemple, la vitre de la serre est brisée par la boule et la
                            boule continue de tomber
   )
   méthodes : association heuristique, propagation de contraintes, raisonnement par défaut, ...;
   connaissances du domaine impliquées : comme ci-dessus, la prédiction s'appliquant à tous
                           les domaines.
```

La structure de tâche ci-dessous ne laisse qu'un degré de liberté dans l'exécution : le nombre d'itérations (récursions) sur les inférences. Voici un exemple de prédiction exhautive.

```
tâche prédiction
  but : effectuer toutes les prédictions relatives dans une certaine situation
  termes de contrôle
     séquence = liste des description-du-système-prédite triée par ordre chronologique
  structure de tâche  prédire (description-du-système -> séquence) {
      spécialiser (... -> description-structurelle-augmentée)
      transformer (... -> description-du-système-prédite
      séquence := ajout en tête de  description-du-système-prédite  à séquence_suite
                    fournie par  prédire (description-du-système-prédite -> séquence_suite)
  }
```

Le niveau stratégique contrôle l'exécution de la structure de tâche en la planifiant puis en surveillant son exécution et, suivant le succès de cette dernière, en mettant à jour (voire en changeant) la-dite structure de tâche. Par exemple, planification de trois niveaux de récursion et bactrack improvisé entre les ST spécialiser et transformer (afin d'obtenir plus de connaissances par plus de spécifications) avant de reprendre comme prévu.
D'autres tâches génériques comme la surveillance ou le diagnostic sont nécessaires pour établir une description complète de la stratégie de raisonnement.

2 *Les tâches de modification de systèmes*

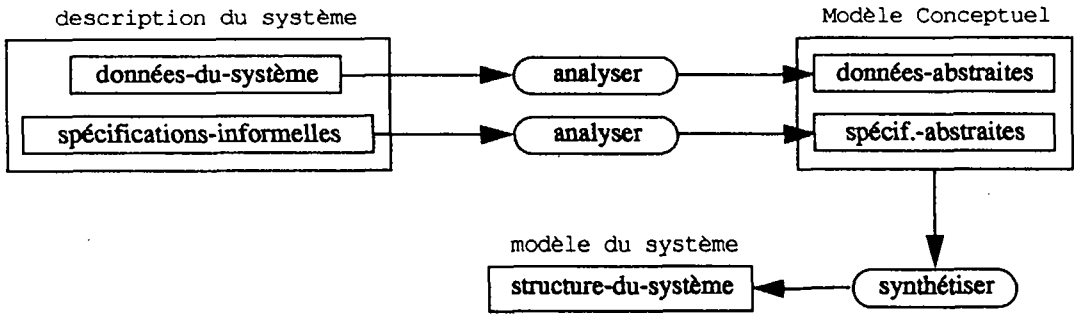
Ce sont des actions modifiant l'état courrant du système, mais ne lui ajoutant pas de nouvelles fonctionnalités. En fait, il s'agit souvent de restaurer les originales. Trois types génériques :

- réparation i.e. le remplacement d'un composant fautif ou suspect à la suite d'un diagnostic ou dans ce dernier, pour tester les composants suspects (c'est une alternative aux mesures). La tâche générique implique la planification, suivant des contraintes physiques ou spatiales, du désassemblage et du réassemblage de parties du système pour remplacer un composant;
- remède i.e. l'initialisation et le contrôle d'un processus neutralisant ou contrebalançant un processu ou un état du système : suppression ou blocage de facteurs causaux, «undo», suppression d'effets indésirables, etc. (il s'agit d'une vue fonctionnelle plus que structurelle);
- contrôle i.e. le maintien de l'intégrité d'un système dynamique; en contrôle de processus, le système est continuement surveillé et toute sortes d'actions «feed-back» sont effectuées (éventuellement des réparations, à titre préventif ou pas, et/ou des remèdes); autre tâche de contrôle très courante, le management de l'information (maintenance de données ou de base de connaissances) contrôle l'exécution de tâches comme la mise à jour, le suivi des versions, l'extension de la structure/organisation, la vérification de la consistence.

Dans des tâches réelles, les tâches de modifications sont ou plus simples ou plus compliquées que leurs formes génériques. Aucun modèle spécifique n'a donc encore été développé. La réparation peut simplement consister à remplacer un composant défectueux par un autre (trivial pour des composants électriques) et le remède à lancer un processus de compensation (refroidissement, administration d'un médicament, etc.).
Inversement, la tâche peut devenir une tâche de synthèse. En effet, si le remplacement d'un toit d'une maison n'exige que de la planification, remplacer ses fondations peut requérir le développement de nouvelles techniques impliquant (localement) de la modélisation, de la conception et de la planification.

3 Les tâches de synthèse

Etant donnée un ensemble de contraintes et de spécifications (souvent informelles), ces tâches synthétisent une structure. L'espace de solution est généralement large, sinon infini. Les modèles d'interprétation associés comportent deux étapes : formalisation des spécifications, création de la structure. Voici un modèle général :



La nature très ouverte des problèmes de synthèse rend les modèles des divers types très différents. Rappelons la taxonomie proposée par KADS :

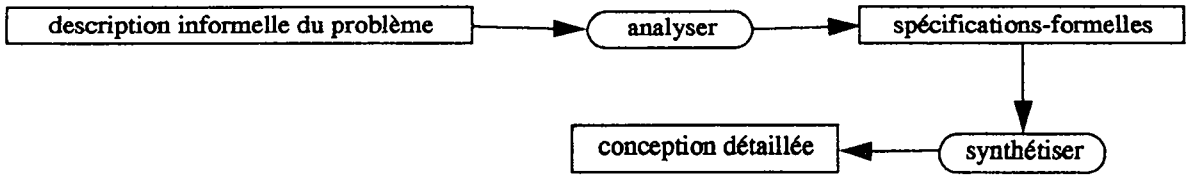
```
transformation
conception
      conception_par_transformation
      conception_par_raffinement
            conception_par_raffinement_à_flot_unique
            conception_par_raffinement_à_flots_multiples
      configuration
planification
modélisation
```

Si de nombreux modèles cognitifs et SE ont été développés pour les tâches d'analyse, très peu le furent pour celles de synthèse. Les modèles présentés plus bas sont donc d'assez haut niveau et ne reflètent qu'une petite partie de la complexité possible des tâches de synthèse réelles. Ainsi, ces tâches réelles ont fréquemment besoin de nombreuses structures intermédiaires, modèles pour le système final mais non formulées dans les termes normalement utilisés pour ce dernier.

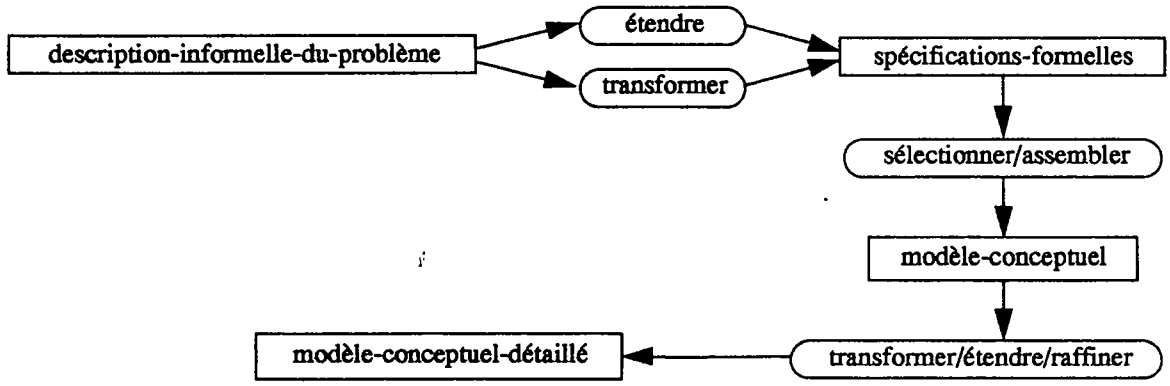
3.1 Conception

Il s'agit de spécifier les composants et la structure d'un produit (objet à fabriquer) compte-tenu de la description (généralement informelle et incomplète) du problème que ce produit doit résoudre. Le processus de conception peut prendre plusieurs formes suivant la complexité du domaine, la standardisation du problème (conception de routine), dans quelle mesure la description du problème définit celui-ci (conception ouverte ou fermée), l'existence de modèles couvrants le cas à résoudre (conception hiérarchique), etc.
Si à une étape de la conception, un modèle (d'une partie) du produit à concevoir est créé et que ce modèle est ensuite raffiné (récursivité du processus) afin d'obtenir le résultat final, on parle de conception hiérarchique. La structure d'inférence de cette phase récursive est présentée plus loin.

La **structure générale des modèles de conception** est la suivante :



Cependant, lorsqu'un Modèle Conceptuel du produit final (spécification de sa structure, quoique dans un langage différent de celui du produit) est impliqué dans le processus de conception, comme c'est généralement le cas, la structure d'inférence de cette tâche peut être représentée ainsi :



Signalons avant la description des ST, que celle des méta-classes est donnée page suivante pour le cas de la «conception à flot unique».

ST **étendre et transformer** //construit une spécification formelle complète du système à élaborer
(in *description-informelle-du-problème* : les paramètres, fonctions et contraintes
(au moins les plus importants);
out *spécifications-formelles* : description dans le langage du domaine de tous les
les paramètres, fonctions et contraintes
)
méthodes : à adapter au domaine ?;
connaissances du domaine impliquées : les fonctions nécessaires dans les systèmes que
l'expert connaît; connaissances sur ce qui est réalisable, etc.

ST **sélectionner/assembler** //i.e. prendre des décisions en fonction des spécifications formelles,
décisions de haut niveau (ex: déterminer le langage dans lequel sera formulé
la conception) comme de bas niveau (ex: éléments spécifiques utilisés)
(in *spécifications-formelles, critères-de-sélection;*
out *modèle-conceptuel* : ses éléments sont structurés par l'assemblage
)
méthodes et connaissances de support : spécifiques au domaine.

```
ST transformer/étendre/raffiner //basé sur une pure méthode de transformation ou sur l'extension
                                des éléments du Modèle Conceptuel ou encore, quand une hiérarchie
                                de squelettes de modèles de conceptions est disponible, un
                                processus de raffinement peut conduire au Modèle Conceptuel
   (in modèle-conceptuel;    out modèle-conceptuel-détaillé;
   )
   méthodes : nombreuses, elles sont soit générales, soit spécifique du domaine;
   connaissances de support : souvent des modèles (partiels) sont utilisés pour guidés le
                              processus de transformation.
```

3.1.1 Conception à flot(s) unique/mutliples

On distingue la conception à flot unique (où une seule structure est construite) de la conception à flots multiples où plusieurs structures ou modèles sont construits concurremment avant que le modèle de conception détaillée soit atteint. Par exemple, la conception d'une usine chimique peut impliquer l'élaboration de différents modèles : chimiques, thermodynamiques et de procédés. Ces modèles de support fournissent des informations et des contraintes sur le flot de conception principal. Plusieurs flots peuvent alors devoir être développés simultanément.

Voici la description des méta-classes impliquées dans la structure d'inférence ci-dessus dans le cas d'un flot unique.

```
rôle description-informelle-du-problème //entrée du processus
   exemples : spécification des E/S, spécification par analogie, description;
   réfère : spécification-informelle-du-système, contraintes, conditions requises.
```

```
rôle spécifications-formelles //résultat de la transformation des specif. informelles en
                              termes de «concepts d'ingénierie» : limites du système, fonctions, buts,
                              critères-de-conception, etc. (cf ci-dessous); c'est le schéma initial dans
                              le langage du domaine
   réfère : critères-de-conception, contraintes-de-conception, variables-de-conception,
            buts-de-conceptions, spécifications-fonctionnelles, spécification-des-composants.
```

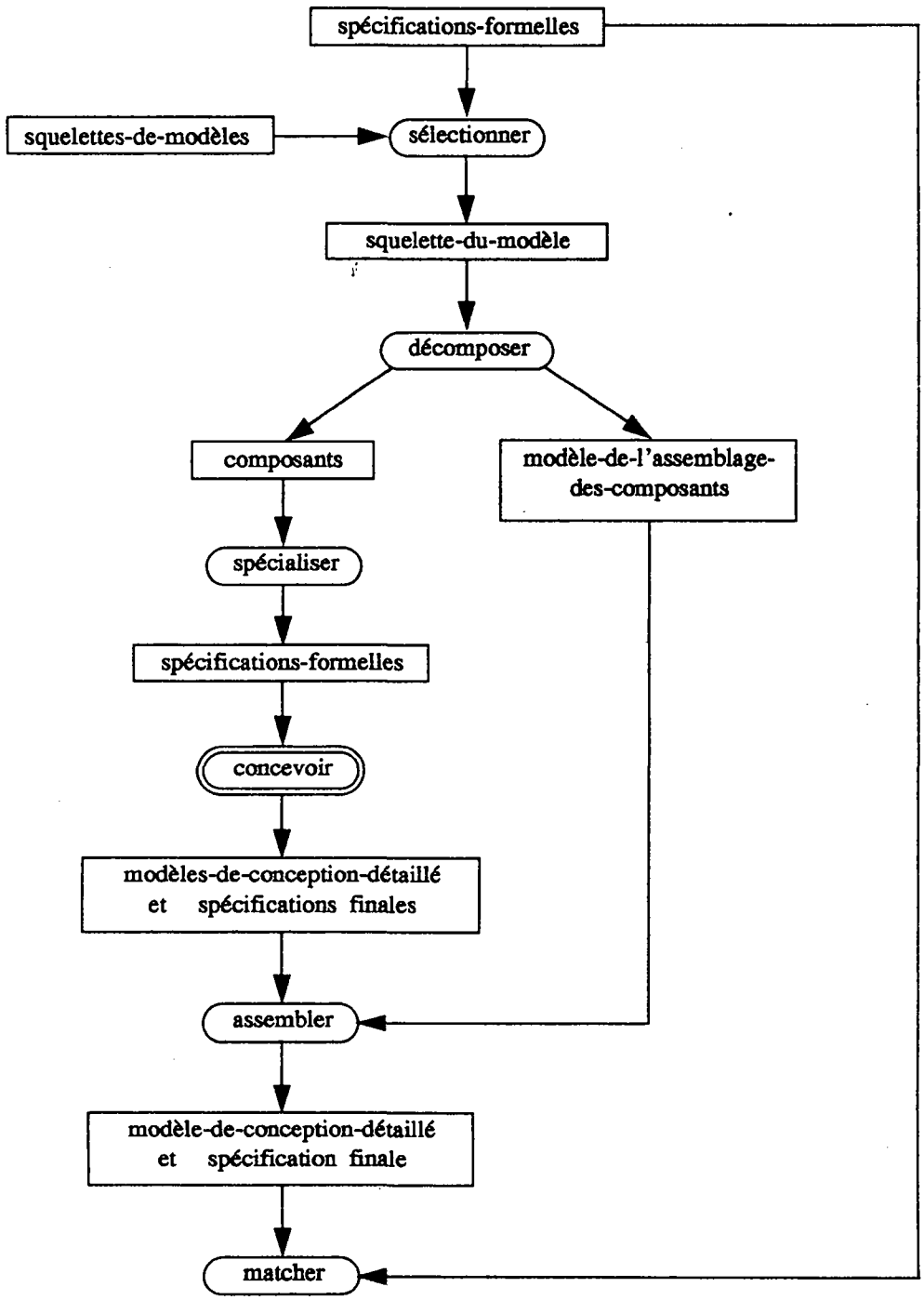
```
rôle modèle-conceptuel //de conception, sa structure est l'abstraction de celle du produit
   exemples : algorithme pour conception de circuits VLSI, description naive d'algorithmes
              d'un programme, description de processus en langage courant, croquis, esquisse;
   réfère : modèle abstrait, caractéristiques du produit, esquisse;
```

```
rôle modèle-conceptuel-détaillé //de l'objet à produire dans tous ses détails; aucune autre
                                spécification ne devrait être nécéssaire pour le construire
   exemples : dessin technique et liste des composants, agencement des composant dans un
              circuit, diagramme électrique;
   réfère : spécification complète des composants, description structurelle complète.
```

3.1.2 Conception hiérarchique

Ce n'est pas un sous-type de tâche mais une variante plus structurée de la structure d'inférence générale pour la conception (et après l'obtention des spécifications formelles), lorsque le domaine est bien connu.

En effet, dans ce cas les problèmes difficiles, comme la décomposition ou bien le développement d'un modèle formel décrivant comment les éléments de l'assemblage interagissent, peuvent être résolus par le remplissage de squelettes de modèles. Une méthode comparable est décrite dans [Chandrasekaran 89] : la «sélection de plans et raffinement». Le squelette de modèle joue ici le rôle de plan et le raffinement consiste en un appel récursif de cette structure d'inférence, via la méta-classe *concevoir*.



ST **sélectionner** //i.e. compare (matche) les spécifications formelles avec les divers squelette de modèles de la base de données et sélectionne le plus approprié
(in *spécifications-formelles* : cf section précédente
modèles : ce n'est pas réellement une méta-classe, c'est une base de données de squelette de modèles, un modèle étant un agrégat de composants et la description des relations entre ceux-ci et l'agrégat; ces relations devraient être suffisantes pour construire une spécification formelle pour chacun des composants, compte-tenu de celles de l'agrégat et inversement pour déduire les spécifications complètes de l'agrégat compte-tenu des spécifications complètes de ses composants;
out *squelette-du-modèle* : un des modèles sélectionné
)
méthodes : association heuristique, satisfaction de contraintes.

ST **décomposer** //le squelette du modèle en divers composants (i.e pour chacun spécifications fonctionnelle mais pas (toutes) les valeurs de paramètres) et un modèle décrivant les relations entre ces composants.

ST **spécialiser** //les spécifications formelles du début et les relations entre le modèle et l'agrégat correspondant pour déduire de nouvelles spécificatins pour chacun des composants; rien de nouveau sur les méta-classes.

ST **concevoir** //appelle la tâche de conception sur chaque composant sauf si leurs spécifications formelles sont déjà du niveau requis pour le résultat de la tâche en cours (description du langage pour des logiciels, composants standard pour la conception d'un circuit); résultat : conception détaillée et spécifications finales pour chaque composant.

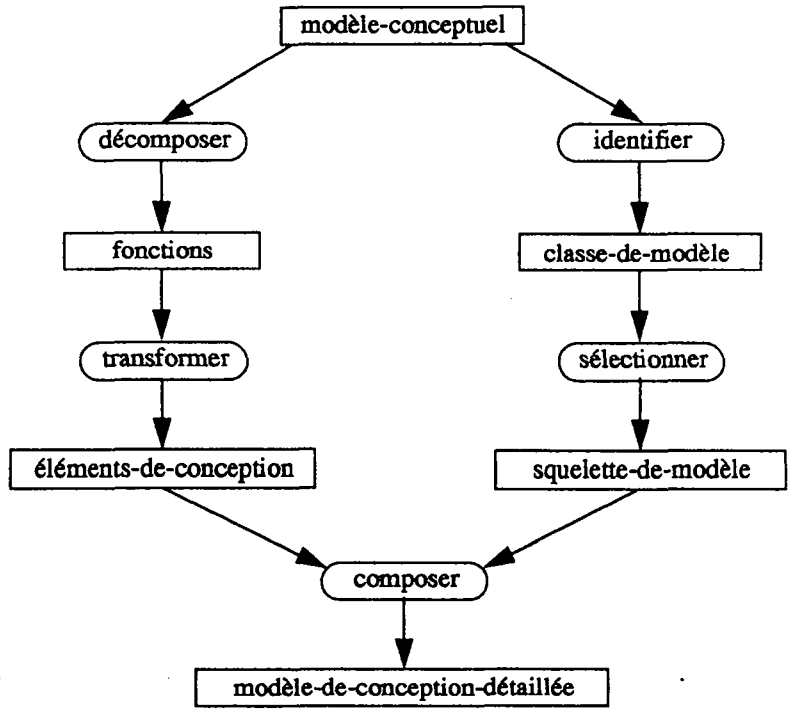
ST **assembler** //les composants conçus en les liant entre eux pour faire un système complet (ex: conception du plan d'un étage, câblage entre les composants sur une carte, etc.); la spécification finale du système est aussi déduite par application des relations entre composants (statisfaction de contraintes) et agrégat sur les spécifications finales des comnposants.

ST **matcher** //les spécifications finales avec les spécifications formelles pour valider la conception.

3.1.3 Conception incrémentale

C'est également une variante plus précise de la structure d'inférence générale pour la conception (après l'obtention du Modèle Conceptuel). Toutes les applications relevant de la conception (ou de la synthèse) peuvent donc s'en inspirer.

Elle ne comporte pas de transformation directe du Modèle Conceptuel vers le modèle de conception détaillée. Les éléments du Modèle Conceptuel sont plutôt transformés individuellement pendant que la structure du modèle de conception détaillée est construite d'après les contraintes et/ou les squelettes de modèle. Voici, dans ce dernier cas, une structure d'inférence montrant cette transformation.



Parfois, la décomposition du Modèle Conceptuel est elle-même dirigée par la structure présente dans un des squelettes de modèle.

3.1.4 Conception par transformation et configuration

La conception par transformation est une forme de conception à flot unique dans laquelle une spécification totale du produit est disponible assez tôt dans le processus de conception mais où dans un formalisme différent de celui de l'implémentation. Le modèle doit donc être transformés une ou plusieurs fois de façon à être implémenté dans les composants du domaine. Un bon exemple est celui de la conception de VLSI où un algorithme (spécification formelle) est en entrée du processus de conception et l'agencement des composants est la sortie désirée.

La configuration est une forme simplifiée de la conception : les composants spécifiques doivent remplir une structure du produit donnée par avance. Par exemple, en «scheduling» (forme de configuration mais analogue à la planification décrite ci-dessous), la structure du plan est donnée mais le remplissage réel par des opérations reste à faire.

3.2 Planification et modélisation

La planification est une forme spécifique de la conception, où les éléments de conception sont des actions et des opérations qui sont combinées dans une séquence de temps pour atteindre un certain but. Il s'agit donc d'une conception dans une dimension - le temps -, avec des contraintes fortes sur le processus de sélection des opérations.

Similaire à la conception, la modélisation a généralement des contraintes plus fortes sur ses entrées : le comportement du produit doit satisfaire certains critères qui sont dérivés de l'objet à modéliser. De tels critères existent rarement en conception. En modélisation, les solutions sont donc plus souvent rejetées ce qui implique de fréquents retours arrières sur les décisions prises afin de créer un modèle adéquat.

Références

1 Intelligence Artificielle - Acquisition des connaissances

[Aussenac 89] N. Aussenac. Conception d'une méthodologie et d'un outil d'acquisition des connaissances expertes. *Thèse de doctorat en Informatique*. Université Paul Sabatier de Toulouse. Octobre 1989.

[Ayel 90] M. Ayel, M.C. Rousset. *La cohérence dans les bases de connaissances, CEPADUES eds.*

[Brachman 85] R. Brachman, J. Schmolze (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science, 9*, pp 171-216.

[Charniak 85] E. Charniak, D. McDermott. Introduction into Artificial Intelligence, *Addison-Wesley*, 1995.

[Clancey 84] W. Clancey, R. Letsinger. NEOMYCIN: Reconfiguring a rulebased expert system for application to teaching. In *W. Clancey & Shortliffe, Eds. Reading in Medical Artificial Intelligence: the first decade, Addison-Wesley*, pp 361-38, 1984.

[Cordier 91] M. O. Cordier, C. Reynaud. Knowledge Acquisition Techniques and Second-Generation Expert Systems. In *Proceedings of the 5th European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW 91)*, Crieff, Scotland, May 1991.

[Dieng 90b] R. Dieng. Méthodes et outils d'acquisition des connaissances. In *ERGO-IA - 90*, pp 245-271, Biarritz, France, Septembre 1990.

[EC2] *Dixièmes Journées Internationales sur les systèmes experts et leurs applications : Catalogue de l'Exposition*. Avignon, Mai-Juin 1990.

[Ganascia 89] J.G. Ganascia. EKAW 90 Tutorial Notes : Machine Learning. In *Proceedings of the 3rd European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW 89)*, pp287--296, Paris, July 1989.

[Gasser 91] L. Gasser. Social conceptions of Knowledge and action: DAI foundations and open system semantics. In *Artificial Intelligence, 47, Special Volume, Foundation of AI*, pp 107-138.

[Gobinet 92] P. Gobinet, K. Causse, D. Cañamero. La généricité en acquisition des connaissances. In *Actes des 3èmes J.A.C. du PRC-IA*, 16-19 avril 1992.

[Goel 87] A. Goel, N. Soundarajan, B. Chandrasekaran. Complexity in classificatory reasonning. In *AAAI-87*, pp 421-425.

[Harmon 85] P. Harmon, D. King. Expert Systems: Artificial Intelligence in Business. *A Wiley Press Book, John Wiley & Sons Inc.*, 1985.

[Hayes-Roth 83] F. Hayes-Roth, D.B. Lenat, D.A. Waterman. Building Expert Systems. *Reading MA:Addison-Wesley Publishing Company*, 1983.

[Hewitt 91] C. Hewitt. Open information System semantics for Distributed Artificial Intelligence. In *Artificial Intelligence, 47, Special Volume, Foundation of AI*, pp 79-106.

[Hiejst 92] G.Van Heijst, P. Terpstra, B.J. Wielinga, N. Shadbolt. Using Generalised Directive Models in Knowledge Acquisition. *Proceeding of the EKAW'92*.

[Hoffman 89] R.R. Hoffman. A survey of methods for Eliciting the Knowledge of Experts. In *SIGART Newsletter, Knowledge Acquisition Special Isse. (108)*:19-27, April 1989.

[Karbach 88] W. Karbach, X. Tong, A. Voss. Filling in the knowledge acquisition gap: via KADS models of expertise to ZDEST2 expert systems. In *proceedings of EKAW'88*, Bonn.

[Karbach 91] W. Karbach, A. Voss, R. Schukey, U. Drouwen. Model-K: Prototyping at the knowledge level. In *proceedings Expert Systems 1991*, Avignon, France, pp 501-512.

[Kodratoff 87] Y. Kodratoff. Leçons d'apprentissage symbolique. *CEPADUES*, 1987.

[Krivine 91] J.P. Krivine, J.M. David. L'acquisition des connaissances vue comme un processus de modélisation : méthodes et outils. *Intellectica*, 1991.

[Kuntzmann 88] A. Kuntzmann-Combelles, C. Vogel. KOD : a support environment for cognitive acquisition and management. In *Safety and Reliability Society Symposium*, 1988.

[Lenat 90] D.B. Lenat, R.V. Guha, K. Pittman, Dexter Pratt, M. Shepherd. Cyc: toward programs with common sense. In *Communications of the ACM*, august 1990.

[Newell 82] A. Newell. The Knowledge Level. In *Artificial Intelligence 18 (1982)*, pp 87-127.

[Pitrat 90] J. Pitrat. *Métaconnaissances*, Hermès, 1990.

[Puerta 91] A. Puerta, J. Edgar, S. Tu, M. Musen. A Multiple-Method Knowledge acquisition shell for the Automatic Generation of Knowledge-Acquisition Tools. In *Proceedings of the 6th. AAAI-KAW*, Banff, Canada, 1991.

[Quinlan 87] J.R. Quinlan, P.J. Compton, K.A. Horn, L. Lazarus. Inductive knowledge acquisition : a case study. In *Artificial Intelligence and Expert Systems*, pp 157--173, 1987.

[Reichgelt 86] H. Reichgelt, F. van Harmelen (1986). Criteria for choosing representation languages and control regimes for expert system. In *Knowledge Engineering Review, 1*, pp 2-17.

[Schank 73] R.C. Schank, K. Colby. *Computer Models of Thought and Language. Freeman*, San Francisco, 1973.

[Skuce 90] D. Skuce, L. Monarch. Ontological Issues in Knowledge Base Design: Some Problems and Suggestions, *Proceeding of thr 5th. AAAI-KAW*, Banff, Canada, 1990, pp 32.1-32.23.

[Schmidt 89] G. Schmidt and T. Wetter. Towards Knowledge Acquisition in Natural Language Dialogue. In *Proceedings of the 3rd European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW 89)*, pages 239--252, Paris, July 1989.

[Steels 90] L. Steels. Component of Expertise. *AI magazine, été 1990*, pp 28-49. Approfondi et complété dans *Steels/Lectures/draft/september-91* sous le titre : Knowledge systems.

[Tong 88] X. Tong, Z. He, R. Yu. A survey of the expert system tool ZDEST-2. In *Proceedings ECAI'88*, Munich, pp 113-118, London: Pitman.

[Vogel 88] C. Vogel. *Génie Cognitif*, Masson, 1988.

2 *Méthodologie KADS*

[Anjewierden 92] A .Anjewierden, J. Wielemaker. Shelley - computer-aided knowledge engineering. In *Knowledge Acquisition (1992) 4*, pp 109-125.

[Breuker 87] Model Driven Knowledge Acquisition: Interpretation Models. Deliverable A1, Esprit Project 1098 Memo 87, VF project Knowledge Acquisition in formal domains. Breuker (ed).

[Breuker 89] Breuker, B. Wielinga. Models of Expertise in Knowledge Acquisition. In Guida and Tasso, editors, *Topics in Expert System Design*, North-Holland, Elsevier Science Publishers B.V., 1989.

[Brunet 91] E. Brunet, C. Toussaint et M. Toyé. Ascopa et Sadse: exemples d'application industrielle de la méthode KADS. In *Avignon-91, Vol. 1*, pp 393-406.

[De Greef 85] P. De Greef, J. Breuker. A case study in structured knowledge acquisition. In *Proc. of the 9th IJCAI*, Los Angeles, CA, pp 390-392, August 1985.

[De Greef 92] P. De Greef, J. Breuker. Analysing system-user cooperation in KADS. In *Knowledge Acquisition (1992) 4*, pp 89-108.

[Harmelen 92] F. van Harmelen, J. Balder. (ML)2: A formal language for KADS models of expertise. *The Knowledge Acquisition Journal*, March 1992.

[Jansweijer 88] W. Jansweijer (1988). *PDP. PhD thesis*, University of Amsterdam.

[Jonker 92] W. Jonker, J.W. Spee. Yet another formalisation of KADS Conceptual Models. *EKAW'92.*

[KADS-II 92Lib] The Common KADS Library, document KADS-II/T1-3/VUB/TR/OO5/1.0.

[Lemmers 89] Lemmers. A shell for systematic diagnosis : structure preserving design of a KBS. *Master's thesis.* University of Amsterdam, Social Science Informatics, 1991

[Schreiber 88] G. Schreiber, J. Breuker, B. Bredeweg, and B. Wielinga. Modelling in KBS Development. In *Proceedings of the 2nd European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW 88)*, pp 7.1-7.15, Bonn, RFA, June 1988.

[Schreiber 89] G. Schreiber, P. De Greef, P. Terpstra, B. Wielinga, E. Brumet, N. Simonin, A. Wallin. A KADS approach to KBS design. ESPRIT project 1098, deliverable B6 UvA-B6-PR-010, University of Amsterdam & Cap Sogeti Innovation.

[Schreiber 91] G. Schreiber, B. Wielinga, and J. Breuker. The KADS Framework for Modelling Expertise. In *Proceedings of the 5th European Workshop on Knowledge Acquisition for Knowledge-Based Systems (EKAW 91)*, Crieff, Scotland, May 1991.

[Schreiber 92] G. Schreiber, B.J. Weilinga, A.Th. Schreiber, H. Akkermans. Differentiating Problem Solving Methods. *Proceedings of the EKAW'92.*

[Voss 90] A. Voss. Model-Based Knowledge Acquisition. In *"Information Systems and Artificial Intelligence: Integration Aspects", SERIES = "Lecture Notes in Computer Science", EDITOR = "D. Karagiannis", PUBLISHER = "Springer-Verlag", VOLUME = 474,* Ulm, FRG, March 1990, pp 256-272.

[Wetter 90] Th. Wetter. First Order Logic Fondation for the KADS Conceptual Model. *EKAW'90.*

[Wetter 91] T. Wetter, W. Schmidt. Formalization of the KADS Interpretation Models. *AISB91.*

[Wielinga 92] B. Wielinga, G. Schreiber, and J. Breuker. KADS: a modelling approach to knowledge engineering. In *Knowledge Acquisition (1992) 4*, pp 136-145

3 Explications

-> AAAI 91: *Proceedings. of the AAAI'91 Workshop on Comparative Analysis of Explanation Planning Architectures*, Anaheim, California, July 1991.

-> EXPL 92: *Actes des deuxièmes journées «Explication» du PRC-GDR IA du CNRS.* Sophia-Antipolis, 17-19 Juin 1992.

[Alvarez 91] I. Alvarez. Explication comparative dans les systèmes experts. In *«Les systèmes experts et leurs applications», Journées Internationales d'Avignon*, mai 1991.

[Appelt 85] D.E. Appelt. Planning Natural Language Utterances. *Cambridge University Press*, Cambridge, U.K., 1985.

[Baker 92] M. Baker. Le rôle de la collaboration dans la construction d'explications. In EXPL 92 pp 25-40.

[Baumewerd 91] A. Baumewerd-Ahlmann, P. Jaschek, J. Kalinski, and H. Lehmkuhl. Embedding Explanations into Model-Based Knowledge Engineering - Improved Decision Support in Environmental Impact Assessment. In *Proceedings of the 1st Int. Conference on Knowledge Modeling and Expertise Transfer*, pp 269-284, Sophia-Antipolis, France, April 1991.

[Bateman 89] J.A. Bateman, C.L. Paris. Phrasing a text in terms the user can understand. In *Procceedings of IJCAI-89*, Detroit, Michigan USA, 20-25 August 1989.

[Bouri 90a] M. Bouri, R. Dieng, G. Kassel and B. Safar. Systèmes à base de connaissances et explications. In *Actes des 3èmes Journées Nationales du PRC-GDR-IA*, pp. 328-339, Paris, Mars 1990.

[Bouri 90b] M. Bouri, R. Dieng, G. Kassel and B. Safar. Vers des systèmes plus explicatifs. In *Actes des 3èmes Journées Nationales du PRC-GDR-IA*, pp. 340-355, Paris, Mars 1990.

[Brézillon 92] P. Brézillon. Intervention de l'utilisateur dans les explications. In EXPL 92 pp 105-130.

[Carcagno 89] D. Carcagno, L. Iordanskaja. Content determination and text structuring in Gossip, *Second European Worshop on Natural Language Generation*, Edingburgh, avril 1989.

[Carcagno 92] D. Carcagno, I. Thomas. Construction d'explications pour un système expert. *Journées Internationales d'Avignon* 1992.

[Cawsey 89] A. Cawsey. Generating Explanatory Discourse: a Plan-Based, Interactive Approach, PhD *Thesis, University of Edinburgh*, 1989.

[Cawsey 91] A. Cawsey. Planning Tailored Interactive Explanations. In AAAI 91 pp 6-14.

[Cawsey 91a] A. Cawsey. Generating Interactive Explanations. *Proceeding of AAAI 91*, Anaheim, CA.

[Chandrasekaran 89] B. Chandrasekaran, M. C. Tanner et J. R. Josephson. Explaining Control Strategies in Problem Solving. In *IEEE Expert (1989)*, pp 9-24.

[Chevallier 92] R. Chevallier. Interactions et explications par la conduite de dialogues coopératifs : l'exemple de STUDIA. In EXPL 92 pp 65-86.

[Clancey 83] W.J. Clancey. The epistemology of a Rule-Based Expert System - A Framework for explanation. In *Artificial Intelligence*, May 1983, pp 215-251.

[Dieng 90a] R. Dieng. Knowledge-based, relation-based and learning-based explanations. In *Proceedings of the 5th Workshop on Explanations*, Manchester, UK, April 1990.

[Dieng 87a] R. Dieng. Un système expert explicateur. In *Journées représentation du raisonnement*, INRIA, Sophia-Antipolis, 22-23 Janvier 1987.

[Dieng 87b] R. Dieng, O. Corby, and P. Haren. Explanatory knowledge tools for expert systems. In D. Sriram and R. A. Adey, editors, *AI in Engineering: Tools and Techniques, Proc. of the 2nd Int. Conf. on Appl. of Art. Int. in Engineering*, pp 320–333, Mech. Comp. Publ., Cambridge, MA, August 1987.

[Dieng 91] R. Dieng, A. Giboin. Expert system explanations: the gap between research and industry. In *Proc. of IJCAI-91 Workshop on Explanation Generation for Knowledge-based Systems*, Sydney, Australia, August 1991.

[Dominguez 89] C.J. Dominguez. Un environnement de développement de systèmes experts explicatifs. In *Journées explications du PRC-IA*, pp 33–49, Gif-sur-Yvette, March 1989.

[Dominguez 92] C.J. Dominguez. L'explicitation des connaissances de contrôle et de stratégie en vue de d'Explication. In EXPL 92 pp 131-146.

[Hasling 83] D.W. Hasling. Abstract explanations of strategy in a diagnostic consultation system. *AAAI-83*, pp 157-167, Washington, august 1983

[Hovy 88] E.H. Hovy. Planning Coherent Multisentential Tests. In *Proceedings of the 26th Annual Meeting of the association of Computational Linguistics*, pp 163-169, June 1988.

[Hugues 87] S. Hugues. Questions classification in rule-based systems. In Research and Developpement in Expert Sytems III, eds Bramer, Cambridge University Press, 1987.

[Gilbert 87] G.N. Gilbert. Question and answer types. In *Expert Systems 87, Cambridge Universty Press.*

[Kassel 86] G. Kassel. Le système d'explication CQFE : une forme de méta-raisonnement intégrant règles et objets. *Thèse de doctorat de l'université de Paris XI*, Orsay, décembre 1896.

[Kassel 92] G. Kassel, M.H. Gréboval, C. Gréboval, F. Bourcier. Raisonner au bon niveau d'abstraction pour produire de meilleures explications: une étude de cas. In EXPL 92 pp 147-161.

[Karsenty 92] L. Karsenty. Les explications spontanées dans des dialogues coopératifs de validation. In EXPL 92 pp 87-102.

[Lambert 87] S. Lambert. An idea for representing strategic knowledge graphically. In *Proceedings of the 3th Alvey Explanation Workshop*, Guildford, UK, pp 94-109, September 1987.

[Lemaire 91] B. Lemaire, B. Safar. Some Necessary Features for Explanation Planning Architectures: Study and Proposal. In AAAI 91 pp 15-26.

[Lemaire 91a] B. Lemaire, B. Safar, C. Yonnet. L'acquisition des connaissances d'explication. In *Actes des 3èmes J.A.C. du PRC-IA*, Dourdan, 14-16 avril 1992.

[Lemaire 92] B. Lemaire. Aspects constructifs de la production d'une explication : l'architecture ESMERALDA. In EXPL 92 pp 165-177.

[Lester 91] J. C. Lester, B. W. Porter. An Architecture for Planning Multi-Paragraph Pedagogical Explanation. In AAAI 91 pp 27-41.

[Maybury 90] M. T. Maybury. Customs Explanations: Exploiting User Models to Plan Multisentential Text. In *Proceedings of the Second International Workshop on User Models*, University of Honolulu, Hawaii, 30 mars -1 avril 1990.

[Maybury 91] M. T. Maybury. An evaluation of Plan-Based Explanation Architecture. In AAAI 91 pp 28-52.

[McCoy 89] K.F. McCoy. Generating context sensitive responses to object-related misconceptions. In *Artificial Intelligence*, Vol. 41, No 2, pp 157-195.

[McKeown 85] K.R. McKeown, M. Wish, K. Matthews. Tailoring explanations for the user. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pp 794-798, Los Angeles, CA, août 1985.

[Mittal 90] V.O. Mittal, C.L. Paris. Analogical explanations in the EES framework. In Proceedings of the 5th Alvey Explanation Workshop, Manshester, UK, April 1990.

[Mooney 91] D. Mooney, S. Carberry, K. McCoy. The Need for Bottom-up Strategies for Generating High-level Structure. In AAAI 91 pp 53-64.

[Moore & Swartout 89] J. D. Moore, W. R. Swartout. A reactive approch to explanation. In *Proceedings of the 11th IJCAI*, Detroit pp 1054-1510, August 1989.

[Moore 91] J. D. Moore, C. L. Paris. The EES Explanation facility: its tasks and its architecture. In AAAI 91 pp 65-79.

[Neches 85] R. Neches, W.R. Swartout, J. Moore. Explainable (and maintainable) expert systems. In *Proc. of the 9th IJCAI*, pp 382-389, Los Angeles, CA, 1985.[Paris 92] C. L. Paris. Systèmes Experts Explicatifs. In EXPL 92 pp 3-23.

[Reynaud 91] C. Reynaud. Là où les problèmes d'acquisition des connaissances et d'explications se rejoignent. In *Proc. of the 1st Int. Conference on Knowledge Modeling and Expertise Transfer*, pp 65–74, Sophia-Antipolis, France, April 1991.

[Safar 89] B. Safar. Répondre à des questions du type Pourquoi pas ?. *Actes des journées Explication, PRC IA*, Gif sur Yvette, mars 1989.

[Safar 92] B. Safar, P. Berthault, J. Sylvestre. Place des explications dans la conception d'une interface intelligente entre une base de données et un usager. In EXPL 92 pp 221-231.

[Sleeman 77] D.H. Sleeman. A system wich allows students to explore algorithms. *Proceedings of the 5th IJCAI*, Cambridge, USA, 1977.

[Sprenger 92] M. Sprenger. Explanations Strategies for KADS-based Expert Systems. *Technical Report 10, Universitat Bielefeld*, GMD, Sankt Augustin, Germany, July 1992.

[Suthers 91] D. D. Suthers. Task-Appropriate Hybrid Architecture for Explanation. In AAAI 91 pp 80-94.

[Swartout 83] W. R. Swartout. XPLAIN: a system for creating and explaining expert consulting systems. *Artificial intelligence*, pp 285–325, September 1983.

[Thomas 92] I. Thomas. Couplage d'un système d'explications à un système expert. In EXPL 92 pp 179-192.

[Weiner 90] J.Weiner. BLAH, a system which explain its reasoning. In *Artificial Intelligence (15)*, pp 19-48.

[Weiner 89] J.Weiner.The effect of User Models on the Production of Explanations. In *Expert Knowledge and Explanation, Charlie Ellis, Ellis Horwood Limited*, pp 144-156, 1989.

[Wick 92] M.R. Wick, W.B. Thomson. Reconstructive Expert System Explanation. *Artificial Intelligence, 54 (1-2)* pp 33-70, Mars 1992.

[Zukerman 91] I. Zukerman. Failure Prevention: an Alternative Approach to Content Planning. In AAAI 91 pp 80-94.

Unité de Recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399